CUARTA EDICIÓN

Redes de computadoras

ANDREW S. TANENBAUM



PEARSON
Prentice Hall

Redes de computadoras

Cuarta edición

Redes de computadoras

Cuarta edición

Andrew S. Tanenbaum

Vrije Universiteit
Amsterdam, The Netherlands

TRADUCCIÓN
Elisa Núñez Ramos
Traductora Profesional

REVISIÓN TÉCNICA
Felipe Antonio Trujillo Fernández
Maestro en Sistemas, Planeación e Informática
Universidad Iberoamericana

Adalberto Francisco García Espinosa
Ingeniero en Sistemas Electrónicos
ITESM–CCM

PEARSON Educación®

México • Argentina • Brasil • Colombia • Costa Rica • Chile • Ecuador
España • Guatemala • Panamá • Perú • Puerto Rico • Uruguay • Venezuela

Datos de catalogación bibliográfica

TANENBAUM, ANDREW S.
Redes de computadoras

PEARSON EDUCACIÓN, México, 2003
ISBN: 970-26-0162-2
Área: Universitarios

Formato 19 × 23.5 cm Páginas: 912

ISBN 0-13-066102-3

Edición en español
Editor: Guillemo Trujano Mendoza
e-mail: guillermo.trujano@pearsoned.com
Editor de desarrollo: Miguel Gutiérrez Hernández
Supervisor de producción: José D. Hernández Garduño

Edición en inglés
Editorial/production supervision: *Patti Guerrieri*
Cover design director: *Jerry Votta*
Cover designer: *Anthony Gemmellaro*
Cover design: *Andrew S. Tanenbaum*
Art director: *Gail Cocker-Bogusz*
Interior Design: *Andrew S. Tanenbaum*
Interior graphics: *Hadel Studio*
Typesetting: *Andrew S. Tanenbaum*
Manufacturing buyer: *Maura Zaldivar*
Executive editor: *Mary Franz*
Editorial assistant: *Noreen Regina*
Marketing manager: *Dan DePasquale*

CUARTA EDICIÓN, 2003

ISBN 970-26-0162-2

Impreso en México. *Printed in Mexico.*

1 2 3 4 5 6 7 8 9 0 - 06 05 04 03

PEARSON
Educación ®

Para Suzanne, Barbara, Marvin y en recuerdo de Bram y Sweetie π

CONTENIDO

PREFACIO

1 INTRODUCCIÓN 1

1.1 USOS DE LAS REDES DE COMPUTADORAS 3
1.1.1 Aplicaciones de negocios 3
1.1.2 Aplicaciones domésticas 6
1.1.3 Usuarios móviles 9
1.1.4 Temas sociales 12

1.2 HARDWARE DE REDES 14
1.2.1 Redes de área local 16
1.2.2 Redes de área metropolitana 18
1.2.3 Redes de área amplia 19
1.2.4 Redes inalámbricas 21
1.2.5 Redes domésticas 23
1.2.6 Interredes 25

1.3 SOFTWARE DE REDES 26
1.3.1 Jerarquías de protocolos 26
1.3.2 Aspectos de diseño de las capas 30
1.3.3 Servicios orientados a la conexión y no orientados a la conexión 32
1.3.4 Primitivas de servicio 34
1.3.5 Relación de servicios a protocolos 36

1.4 MODELOS DE REFERENCIA 37
1.4.1 El modelo de referencia OSI 37
1.4.2 El modelo de referencia TCP/IP 41
1.4.3 Comparación entre los modelos de referencia OSI y TCP/IP 44
1.4.4 Crítica al modelo OSI y los protocolos 46
1.4.5 Crítica del modelo de referencia TCP/IP 48

1.5 REDES DE EJEMPLO 49
1.5.1 Internet 50
1.5.2 Redes orientadas a la conexión: X.25, Frame Relay y ATM 59
1.5.3 Ethernet 65
1.5.4 LANs inalámbricas: 802.11 68

1.6 ESTANDARIZACIÓN DE REDES 71
1.6.1 Quién es quién en el mundo de las telecomunicaciones 71
1.6.2 Quién es quién en los estándares internacionales 74
1.6.3 Quién es quién en el mundo de los estándares de Internet 75

1.7 UNIDADES MÉTRICAS 77

1.8 PANORAMA DEL RESTO DEL LIBRO 78

1.9 RESUMEN 80

2 LA CAPA FÍSICA 85

2.1 LA BASE TEÓRICA DE LA COMUNICACIÓN DE DATOS 85
2.1.1 El análisis de Fourier 86
2.1.2 Señales de ancho de banda limitado 86
2.1.3 La tasa de datos máxima de un canal 89

2.2 MEDIOS DE TRANSMISIÓN GUIADOS 90
2.2.1 Medios magnéticos 90
2.2.2 Par trenzado 91
2.2.3 Cable coaxial 92
2.2.4 Fibra óptica 93

2.3 TRANSMISIÓN INALÁMBRICA 100
2.3.1 El espectro electromagnético 100
2.3.2 Radiotransmisión 103

2.3.3 Transmisión por microondas 104
2.3.4 Ondas infrarrojas y milimétricas 106
2.3.5 Transmisión por ondas de luz 107

2.4 SATÉLITES DE COMUNICACIONES 109
2.4.1 Satélites geoestacionarios 109
2.4.2 Satélites de Órbita Terrestre Media 113
2.4.3 Satélites de Órbita Terrestre Baja 114
2.4.4 Satélites en comparación con fibra óptica 117

2.5 LA RED TELEFÓNICA PÚBLICA CONMUTADA 118
2.5.1 Estructura del sistema telefónico 119
2.5.2 La política de los teléfonos 122
2.5.3 El circuito local: módems, ADSL e inalámbrico 124
2.5.4 Troncales y multiplexión 137
2.5.5 Conmutación 146

2.6 EL SISTEMA TELEFÓNICO MÓVIL 152
2.6.1 Teléfonos móviles de primera generación 153
2.6.2 Teléfonos móviles de segunda generación: voz digital 157
2.6.3 Teléfonos móviles de tercera generación: voz y datos digitales 166

2.7 TELEVISIÓN POR CABLE 169
2.7.1 Televisión por antena comunal 169
2.7.2 Internet a través de cable 170
2.7.3 Asignación de espectro 172
2.7.4 Módems de cable 173
2.7.5 ADSL en comparación con el cable 175

2.8 RESUMEN 177

3 LA CAPA DE ENLACE DE DATOS 183

3.1 CUESTIONES DE DISEÑO DE LA CAPA DE ENLACE DE DATOS 184
3.1.1 Servicios proporcionados a la capa de red 184
3.1.2 Entramado 187
3.1.3 Control de errores 191
3.1.4 Control de flujo 192

3.2 DETECCIÓN Y CORRECCIÓN DE ERRORES 192
3.2.1 Códigos de corrección de errores 193
3.2.2 Códigos de detección de errores 196

3.3 PROTOCOLOS ELEMENTALES DE ENLACE DE DATOS 200
3.3.1 Un protocolo símplex sin restricciones 204
3.3.2 Protocolo símplex de parada y espera 206
3.3.3 Protocolo símplex para un canal con ruido 208

3.4 PROTOCOLOS DE VENTANA CORREDIZA 211
3.4.1 Un protocolo de ventana corrediza de un bit 214
3.4.2 Protocolo que usa retroceso N 216
3.4.3 Protocolo que utiliza repetición selectiva 223

3.5 VERIFICACIÓN DE LOS PROTOCOLOS 229
3.5.1 Modelos de máquinas de estado finito 229
3.5.2 Modelos de red de Petri 232

3.6 EJEMPLOS DE PROTOCOLOS DE ENLACE DE DATOS 234
3.6.1 HDLC—Control de Enlace de Datos de Alto Nivel 234
3.6.2 La capa de enlace de datos en Internet 237

3.7 RESUMEN 242

4 LA SUBCAPA DE CONTROL DE ACCESO AL MEDIO 247

4.1 EL PROBLEMA DE ASIGNACIÓN DEL CANAL 248
4.1.1 Asignación estática de canal en LANs y MANs 248
4.1.2 Asignación dinámica de canales en LANs y MANs 249

4.2 PROTOCOLOS DE ACCESO MÚLTIPLE 251
4.2.1 ALOHA 251
4.2.2 Protocolos de acceso múltiple con detección de portadora 255
4.2.3 Protocolos libres de colisiones 259
4.2.4 Protocolos de contención limitada 261
4.2.5 Protocolos de acceso múltiple por división de longitud de onda 265
4.2.6 Protocolos de LANs inalámbricas 267

4.3 ETHERNET 271
4.3.1 Cableado Ethernet 271
4.3.2 Codificación Manchester 274
4.3.3 El protocolo de subcapa MAC de Ethernet 275
4.3.4 Algoritmo de retroceso exponencial binario 278
4.3.5 Desempeño de Ethernet 279
4.3.6 Ethernet conmutada 281
4.3.7 Fast Ethernet 283
4.3.8 Gigabit Ethernet 286
4.3.9 Estándar IEEE 802.2: control lógico del enlace 290
4.3.10 Retrospectiva de Ethernet 291

4.4 LANS INALÁMBRICAS 292
4.4.1 La pila de protocolos del 802.11 292
4.4.2 La capa física del 802.11 293
4.4.3 El protocolo de la subcapa MAC del 802.11 295
4.4.4 La estructura de trama 802.11 299
4.4.5 Servicios 301

4.5 BANDA ANCHA INALÁMBRICA 302
4.5.1 Comparación entre los estándares 802.11 y 802.16 303
4.5.2 La pila de protocolos del estándar 802.16 305
4.5.3 La capa física del estándar 802.16 306
4.5.4 El protocolo de la subcapa MAC del 802.16 307
4.5.5 La estructura de trama 802.16 309

4.6 BLUETOOTH 310
4.6.1 Arquitectura de Bluetooth 311
4.6.2 Aplicaciones de Bluetooth 312
4.6.3 La pila de protocolos de Bluetooth 313
4.6.4 La capa de radio de Bluetooth 314
4.6.5 La capa de banda base de Bluetooth 315
4.6.6 La capa L2CAP de Bluetooth 316
4.6.7 Estructura de la trama de Bluetooth 316

4.7 CONMUTACIÓN EN LA CAPA DE ENLACE DE DATOS 317
4.7.1 Puentes de 802.x a 802.y 319
4.7.2 Interconectividad local 322
4.7.3 Puentes con árbol de expansión 323
4.7.4 Puentes remotos 325
4.7.5 Repetidores, concentradores, puentes, conmutadores, enrutadores y puertas de enlace 326
4.7.6 LANs virtuales 328

4.8 RESUMEN 336

5 LA CAPA DE RED 343

5.1 ASPECTOS DE DISEÑO DE LA CAPA DE RED 343
- 5.1.1 Conmutación de paquetes de almacenamiento y reenvío 344
- 5.1.2 Servicios proporcionados a la capa de transporte 344
- 5.1.3 Implementación del servicio no orientado a la conexión 345
- 5.1.4 Implementación del servicio orientado a la conexión 347
- 5.1.5 Comparación entre las subredes de circuitos virtuales y las de datagramas 348

5.2 ALGORITMOS DE ENRUTAMIENTO 350
- 5.2.1 Principio de optimización 352
- 5.2.2 Enrutamiento por la ruta más corta 353
- 5.2.3 Inundación 355
- 5.2.4 Enrutamiento por vector de distancia 357
- 5.2.5 Enrutamiento por estado del enlace 360
- 5.2.6 Enrutamiento jerárquico 366
- 5.2.7 Enrutamiento por difusión 368
- 5.2.8 Enrutamiento por multidifusión 370
- 5.2.9 Enrutamiento para *hosts* móviles 372
- 5.2.10 Enrutamiento en redes *ad hoc* 375
- 5.2.11 Búsqueda de nodos en redes de igual a igual 380

5.3 ALGORITMOS DE CONTROL DE CONGESTIÓN 384
- 5.3.1 Principios generales del control de congestión 386
- 5.3.2 Políticas de prevención de congestión 388
- 5.3.3 Control de congestión en subredes de circuitos virtuales 389
- 5.3.4 Control de congestión en subredes de datagramas 391
- 5.3.5 Desprendimiento de carga 394
- 5.3.6 Control de fluctuación 395

5.4 CALIDAD DEL SERVICIO 397
- 5.4.1 Requerimientos 397
- 5.4.2 Técnicas para alcanzar buena calidad de servicio 398
- 5.4.3 Servicios integrados 409
- 5.4.4 Servicios diferenciados 412
- 5.4.5 Conmutación de etiquetas y MPLS 415

5.5 INTERCONECTIVIDAD 418
- 5.5.1 Cómo difieren las redes 419
- 5.5.2 Conexión de redes 420
- 5.5.3 Circuitos virtuales concatenados 422
- 5.5.4 Interconectividad no orientada a la conexión 423

5.5.5 Entunelamiento 425
5.5.6 Enrutamiento entre redes 426
5.5.7 Fragmentación 427

5.6 LA CAPA DE RED DE INTERNET 431
5.6.1 El protocolo IP 433
5.6.2 Direcciones IP 436
5.6.3 Protocolos de Control en Internet 449
5.6.4 OSPF—Protocolos de Enrutamiento de Puerta de Enlace Interior 454
5.6.5 BGP—Protocolo de Puerta de Enlace de Frontera 459
5.6.6 Multidifusión de Internet 461
5.6.7 IP móvil 462
5.6.8 IPv6 464

5.7 RESUMEN 473

6 LA CAPA DE TRANSPORTE 481

6.1 EL SERVICIO DE TRANSPORTE 481
6.1.1 Servicios proporcionados a las capas superiores 481
6.1.2 Primitivas del servicio de transporte 483
6.1.3 *Sockets* de Berkeley 487
6.1.4 Un ejemplo de programación de *sockets*: un servidor de archivos de Internet 488

6.2 ELEMENTOS DE LOS PROTOCOLOS DE TRANSPORTE 492
6.2.1 Direccionamiento 493
6.2.2 Establecimiento de una conexión 496
6.2.3 Liberación de una conexión 502
6.2.4 Control de flujo y almacenamiento en búfer 506
6.2.5 Multiplexión 510
6.2.6 Recuperación de caídas 511

6.3 UN PROTOCOLO DE TRANSPORTE SENCILLO 513
6.3.1 Las primitivas de servicio de ejemplo 513
6.3.2 La entidad de transporte de ejemplo 515
6.3.3 El ejemplo como máquina de estados finitos 522

6.4 LOS PROTOCOLOS DE TRANSPORTE DE INTERNET: UDP 524
6.4.1 Introducción a UDP 525
6.4.2 Llamada a procedimiento remoto 526
6.4.3 El protocolo de transporte en tiempo real 529

6.5 LOS PROTOCOLOS DE TRANSPORTE DE INTERNET: TCP 532
6.5.1 Introducción a TCP 532
6.5.2 El modelo del servicio TCP 533
6.5.3 El protocolo TCP 535
6.5.4 El encabezado del segmento TCP 536
6.5.5 Establecimiento de una conexión TCP 539
6.5.6 Liberación de una conexión TCP 541
6.5.7 Modelado de administración de conexiones TCP 541
6.5.8 Política de transmisión del TCP 543
6.5.9 Control de congestión en TCP 547
6.5.10 Administración de temporizadores del TCP 550
6.5.11 TCP y UDP inalámbricos 553
6.5.12 TCP para Transacciones 555

6.6 ASPECTOS DEL DESEMPEÑO 557
6.6.1 Problemas de desempeño en las redes de cómputo 557
6.6.2 Medición del desempeño de las redes 560
6.6.3 Diseño de sistemas para un mejor desempeño 562
6.6.4 Procesamiento rápido de las TPDUs 566
6.6.5 Protocolos para redes de gigabits 569

6.7 RESUMEN 573

7 LA CAPA DE APLICACIÓN 579

7.1 DNS—EL SISTEMA DE NOMBRES DE DOMINIO 579
7.1.1 El espacio de nombres del DNS 580
7.1.2 Registros de recursos 582
7.1.3 Servidores de nombres 586

7.2 CORREO ELECTRÓNICO 588
7.2.1 Arquitectura y servicios 590
7.2.2 El agente de usuario 591
7.2.3 Formatos de mensaje 594
7.2.4 Transferencia de mensajes 602
7.2.5 Entrega final 605

7.3 WORLD WIDE WEB 611
7.3.1 Panorama de la arquitectura 612
7.3.2 Documentos Web estáticos 629

7.3.3 Documentos Web dinámicos 643
7.3.4 HTTP—Protocolo de Transferencia de Hipertexto 651
7.3.5 Mejoras de desempeño 656
7.3.6 La Web inalámbrica 662

7.4 MULTIMEDIA 674
7.4.1 Introducción al audio digital 674
7.4.2 Compresión de audio 676
7.4.3 Audio de flujo continuo 679
7.4.4 Radio en Internet 683
7.4.5 Voz sobre IP 685
7.4.6 Introducción al vídeo 692
7.4.7 Compresión de vídeo 696
7.4.8 Vídeo bajo demanda 704
7.4.9 Mbone—Red dorsal de multidifusión 711

7.5 RESUMEN 714

8 SEGURIDAD EN REDES 721

8.1 CRIPTOGRAFÍA 724
8.1.1 Introducción a la criptografía 725
8.1.2 Cifrados por sustitución 727
8.1.3 Cifrados por transposición 729
8.1.4 Rellenos de una sola vez 730
8.1.5 Dos principios criptográficos fundamentales 735

8.2 ALGORITMOS DE CLAVE SIMÉTRICA 737
8.2.1 DES—El Estándar de Encriptación de Datos 738
8.2.2 AES—El Estándar de Encriptación Avanzada 741
8.2.3 Modos de cifrado 745
8.2.4 Otros cifrados 750
8.2.5 Criptoanálisis 750

8.3 ALGORITMOS DE CLAVE PÚBLICA 752
8.3.1 El algoritmo RSA 753
8.3.2 Otros algoritmos de clave pública 755

8.4 FIRMAS DIGITALES 755
8.4.1 Firmas de clave simétrica 756
8.4.2 Firmas de clave pública 757
8.4.3 Compendios de mensaje 759
8.4.4 El ataque de cumpleaños 763

8.5 ADMINISTRACIÓN DE CLAVES PÚBLICAS 765
8.5.1 Certificados 765
8.5.2 X.509 767
8.5.3 Infraestructuras de clave pública 768

8.6 SEGURIDAD EN LA COMUNICACIÓN 772
8.6.1 Ipsec 772
8.6.2 *Firewalls* 776
8.6.3 Redes privadas virtuales 779
8.6.4 Seguridad inalámbrica 780

8.7 PROTOCOLOS DE AUTENTICACIÓN 785
8.7.1 Autenticación basada en una clave secreta compartida 786
8.7.2 Establecimiento de una clave compartida: el intercambio de claves de Diffie-Hellman 791
8.7.3 Autenticación que utiliza un centro de distribución de claves 793
8.7.4 Autenticación utilizando Kerberos 796
8.7.5 Autenticación utilizando criptografía de clave pública 798

8.8 SEGURIDAD DE CORREO ELECTRÓNICO 799
8.8.1 PGP—Privacidad Bastante Buena 799
8.8.2 PEM—Correo con Privacidad Mejorada 803
8.8.3 S/MIME 804

8.9 SEGURIDAD EN WEB 805
8.9.1 Amenazas 805
8.9.2 Asignación segura de nombres 806
8.9.3 SSL—La Capa de Sockets Seguros 813
8.9.4 Seguridad de código móvil 816

8.10 ASPECTOS SOCIALES 819
8.10.1 Privacidad 819
8.10.2 Libertad de expresión 822
8.10.3 Derechos de autor 826

8.11 RESUMEN 828

9 LISTA DE LECTURAS Y BIBLIOGRAFÍA 835

9.1. SUGERENCIAS DE LECTURAS ADICIONALES 835
- 9.1.1 Introducción y obras generales 836
- 9.1.2 La capa física 838
- 9.1.3 La capa de enlace de datos 840
- 9.1.4 La subcapa de control de acceso al medio 840
- 9.1.5 La capa de red 842
- 9.1.6 La capa de transporte 844
- 9.1.7 La capa de aplicación 844
- 9.1.8 Seguridad en redes 846

9.2 BIBLIOGRAFÍA EN ORDEN ALFABÉTICO 848

ÍNDICE 869

PREFACIO

La presente es la cuarta edición de este libro. Cada edición ha correspondido a una fase diferente de la manera en que se usaron las redes de computadoras. Cuando apareció la primera edición, en 1980, las redes eran una curiosidad académica. Para la segunda edición, en 1988, las redes ya se usaban en universidades y en grandes empresas. Y en 1996, cuando se editó por tercera vez este libro, las redes de computadoras, en particular Internet, se habían convertido en una realidad cotidiana para millones de personas. El elemento nuevo de la cuarta edición es el rápido crecimiento de las redes inalámbricas en muchas formas.

El panorama de las redes ha cambiado radicalmente desde la tercera edición. A mediados de la década de 1990 existían varios tipos de LANs y WANs, junto con pilas de múltiples protocolos. Para el 2003, la única LAN alámbrica de amplio uso tal vez sea Ethernet y prácticamente todas las WANs estarían en Internet. En consecuencia, se habrá eliminado una gran cantidad de material referente a estas antiguas redes.

Sin embargo, también abundan los nuevos desarrollos. Lo más importante es el gran aumento de redes inalámbricas, como la 802.11, los ciclos locales inalámbricos, las redes celulares 2G y 3G, Bluetooth, WAP (protocolo de aplicaciones inalámbricas), el i-mode y otros. De acuerdo con esto, se ha agregado una gran cantidad de material a las redes inalámbricas. Otro tema importante y novedoso es la seguridad, por lo que se ha agregado todo un capítulo al respecto.

Aun cuando el capítulo 1 tiene la misma función introductoria que en la tercera edición, su contenido se ha revisado y actualizado. Por ejemplo, en dicho capítulo se presentan introducciones a Internet, Ethernet y LANs inalámbricas, además de algunas referencias y datos históricos. También se explican brevemente las redes domésticas.

El capítulo 2 se ha reorganizado. Luego de una breve introducción a los principios de comunicación de datos, hay tres secciones importantes sobre la transmisión (medios guiados, inalámbricos y por satélite), seguidas de otras tres con ejemplos importantes (el sistema público de telefonía conmutada, el sistema de teléfonos celulares y la TV por cable). Entre los nuevos temas tratados en este capítulo están ADSL, banda ancha inalámbrica, MANs inalámbricas y acceso a Internet por cable y DOCSIS.

El capítulo 3 siempre ha presentado los principios fundamentales de los protocolos de punto a punto. Estas ideas son permanentes y no han cambiado durante décadas.

En consecuencia, las series de protocolos de ejemplo detallados que se presentan en este capítulo no han cambiado en lo más mínimo desde la tercera edición.

En contraste, la subcapa MAC ha sido un área de gran actividad en los últimos años, por lo que se presentan muchos cambios en el capítulo 4. La sección sobre Ethernet se ha ampliado para incluir la Ethernet de gigabits. Las secciones nuevas importantes son las que tratan sobre las LANs inalámbricas, banda ancha inalámbrica, Bluetooth y la conmutación de la capa de enlace de datos, incluyendo MPLS.

También se actualizó el capítulo 5, en donde se eliminó todo lo referente a ATM y se incluyó material adicional sobre Internet.

Ahora la calidad del servicio también es un tema importante, incluyendo las exposiciones de los servicios integrados y los servicios diferenciados. Las redes inalámbricas también están presentes aquí, con una explicación del enrutamiento en redes *ad hoc*. Entre otros temas nuevos se encuentran las redes NAT y de igual a igual.

El capítulo 6 trata aún de la capa de transporte, pero aquí también se han hecho algunos cambios, que incluyen un ejemplo de la programación de *sockets*. También se explican un cliente y un servidor de una página en C. Estos programas, disponibles en el sitio Web del libro, se pueden compilar y ejecutar. En conjunto proporcionan un servidor Web de archivos remoto para experimentación. Entre otros temas están la llamada a procedimiento remoto, RTP y el TCP para transacciones.

El capítulo 7 se ha enfocado sobre todo en la capa de aplicación. Después de una breve introducción sobre DNS, el resto del capítulo aborda sólo tres temas: el correo electrónico, Web y multimedia. Pero cada tema se trata con todo detalle. La exposición de cómo funciona Web abarca ahora más de 60 páginas, y cubre una amplia serie de temas, entre ellos las páginas Web estáticas y dinámicas, HTTP, los scripts (secuencias de comandos) de CGI, redes de distribución de información, *cookies* y el uso de caché en Web. También se incluye material sobre cómo se escriben las páginas Web modernas, incluyendo breves introducciones a XML, XSL, XHTML, PHP y más; todo con ejemplos que se pueden probar. Asimismo, hay una exposición sobre Web inalámbrica, enfocándose en i-mode y WAP. El material de multimedia incluye ahora MP3, audio de flujo continuo, radio en Internet y voz sobre IP.

La seguridad ha llegado a ser tan importante que ahora se ha ampliado a un capítulo entero de más de 100 páginas (el capítulo 8). Cubre tanto los principios de la seguridad (algoritmos simétricos y de clave pública, firmas digitales y certificados X.509) como las aplicaciones de estos principios (autenticación, seguridad del correo electrónico y seguridad en Web). El capítulo es amplio (va desde la criptografía cuántica hasta la censura gubernamental) y profundo (por ejemplo, trata en detalle el funcionamiento de SHA-1).

El capítulo 9 contiene una lista totalmente nueva de lecturas sugeridas y una amplia bibliografía de más de 350 citas a la literatura actual. Más de 200 de éstas son a artículos y libros escritos en el 2000 o más recientes.

Los libros de computación están llenos de acrónimos, y éste no es la excepción. Para cuando acabe de leer el presente libro, los siguientes términos le serán familiares: ADSL, AES, AMPS, AODV, ARP, ATM, BGP, CDMA, CDN, CGI, CIDR, DCF, DES, DHCP, DMCA, FDM, FHSS, GPRS, GSM, HDLC, HFC, HTML, HTTP, ICMP, IMAP, ISP, ITU, LAN, LMDS, MAC, MACA, MIME, MPEG, MPLS, MTU, NAP, NAT, NSA, NTSC, OFDM, OSPF, PCF, PCM, PGP, PHP, PKI,

POTS, PPP, PSTN, QAM, QPSK, RED, RFC, RPC, RSA, RSVP, RTP, SSL, TCP, TDM, UDP, URL, UTP, VLAN, VPN, VSAT, WAN, WAP, WDMA, WEP, WWW y XML. Pero no se preocupe. Cada uno se definirá cuidadosamente antes de usarlo.

Para ayudar a los profesores a utilizar este libro como texto en un curso, el autor ha preparado varios apoyos, en inglés, para la enseñanza, como:

- Un manual de solución de problemas.
- Archivos que contienen las figuras del libro en varios formatos.
- Páginas de PowerPoint para un curso que utilice el libro.
- Un simulador (escrito en C) para los protocolos de ejemplo del capítulo 3.
- Una página Web con vínculos a muchos manuales en línea, empresas, listas de preguntas frecuentes, etcétera.

El manual de soluciones sólo podrá adquirirlo directamente con los representantes de Pearson Educación (pero **sólo** está disponible para los profesores; los estudiantes no podrán adquirirlo). El resto del material está en el sitio Web del libro:

http://www.pearsonedlatino.com/tanenbaum

Localice la portada del libro y haga clic en ella.

Muchas personas me ayudaron durante la preparación de la cuarta edición. Me gustaría agradecer especialmente a: Ross Anderson, Elizabeth Belding Royer, Steve Bellovin, Chatschick Bisdikian, Kees Bot, Scott Bradner, Jennifer Bray, Pat Cain, Ed Felten, Warwick Ford, Kevin Fu, Ron Fulle, Jim Geier, Mario Gerla, Natalie Giroux, Steve Hanna, Jeff Hayes, Amir Herzberg, Philip Homburg, Philipp Hoschka, David Green, Bart Jacobs, Frans Kaashoek, Steve Kent, Roger Kermode, Robert Kinicki, Shay Kutten, Rob Lanphier, Marcus Leech, Tom Maufer, Brent Miller, Shivakant Mishra, Thomas Nadeau, Shlomo Ovadia, Kaveh Pahlavan, Radia Perlman, Guillaume Pierre, Wayne Pleasant, Patrick Powell, Tomas Robertazzi, Medy Sanadidi, Christian Schmutzer, Henning Schulzrinne, Paul Sevinc, Mihail Sichitiu, Bernard Sklar, Ed Skoudis, Bob Strader, George Swallow, George Thiruvathukal, Peter Tomsu, Patrick Verkaik, Dave Vittali, Spyros Voulgaris, Jan-Mark Wams, Ruediger Weis, Bert Wijnen, Joseph Wilkes, Leendert van Doorn y Maarten van Steen.

Un especial agradecimiento a Trudy Levine por demostrar que las abuelas pueden hacer un trabajo fino de revisión de material técnico. Shivakant Mishra ideó muchos de los desafiantes problemas de fin de capítulo. Andy Dornan sugirió lecturas adicionales para el capítulo 9. Jan Looyen proporcionó hardware esencial en un momento crítico. El doctor F. de Nies hizo un excelente trabajo de cortado y pegado justo cuando fue necesario. Mary Franz, mi editora en Prentice Hall, me proporcionó más material de lectura del que había consumido en los 7 años anteriores y fue de gran ayuda en muchos otros aspectos.

Finalmente, llegamos a las personas más importantes: Suzanne, Barbara y Marvin. A Suzanne por su amor, paciencia y los almuerzos. A Barbara y Marvin por ser agradables y joviales (excepto al quejarse de los horribles libros de texto universitarios, manteniéndome, de este modo, alerta). Gracias.

ANDREW S. TANENBAUM

1

INTRODUCCIÓN

Cada uno de los tres últimos siglos fue dominado por una tecnología. El siglo XVIII fue la era de los grandes sistemas mecánicos que acompañaron la Revolución Industrial. El siglo XIX fue la edad de la máquina de vapor. Durante el siglo XX la tecnología clave fue la obtención, el procesamiento y la distribución de la información. Entre otros acontecimientos, vimos la instalación de redes mundiales de telefonía, la invención de la radio y la televisión, el nacimiento y crecimiento sin precedentes de la industria de la computación, así como el lanzamiento de satélites de comunicaciones.

Como resultado del rápido progreso tecnológico, estas áreas están convergiendo de una manera acelerada y las diferencias entre la recolección, transportación, almacenamiento y procesamiento de la información están desapareciendo rápidamente. Organizaciones con cientos de oficinas dispersas en una amplia área geográfica esperan de manera rutinaria poder examinar el estado actual incluso de su sucursal más distante con sólo oprimir un botón. Al aumentar nuestra capacidad de obtener, procesar y distribuir información, la demanda de procesamiento de información cada vez más complejo crece incluso con más celeridad.

Aunque la industria de la computación aún es joven en comparación con otras industrias (como la automotriz y la aeronáutica), ha progresado espectacularmente en poco tiempo. Durante las dos primeras décadas de su existencia, los sistemas de computación estaban altamente centralizados, por lo general, en una sala grande e independiente. Con frecuencia, estas salas tenían paredes de cristal a través de las cuales los visitantes podían atisbar la maravilla electrónica que encerraban. Las compañías o universidades medianas apenas llegaban a tener una o dos computadoras, en tanto que las

instituciones grandes tenían, cuando mucho, una docena. La idea de que en veinte años se pudieran producir en masa millones de computadoras igualmente poderosas pero más pequeñas que un timbre postal era ciencia-ficción.

La fusión de las computadoras y las comunicaciones ha tenido una influencia profunda en la manera en que están organizados los sistemas computacionales. Actualmente, el concepto de "centro de cómputo" como un espacio amplio con una computadora grande a la que los usuarios llevaban su trabajo a procesar es totalmente obsoleto. El modelo antiguo de una sola computadora que realiza todas las tareas computacionales de una empresa ha sido reemplazado por otro en el que un gran número de computadoras separadas pero interconectadas hacen el trabajo. Estos sistemas se denominan **redes de computadoras.** El diseño y la organización de estas redes es el objetivo de este libro.

A lo largo del libro utilizaremos el término "redes de computadoras" para designar un conjunto de computadoras autónomas interconectadas. Se dice que dos computadoras están interconectadas si pueden intercambiar información. No es necesario que la conexión se realice mediante un cable de cobre; también se pueden utilizar las fibras ópticas, las microondas, los rayos infrarrojos y los satélites de comunicaciones. Las redes tienen varios tamaños, formas y figuras, como veremos más adelante. Aunque a algunas personas les parezca extraño, ni Internet ni Web son una red de computadoras. Este concepto quedará claro al finalizar el libro. La respuesta rápida es: Internet no es una red única, sino una red de redes, y Web es un sistema distribuido que se ejecuta sobre Internet.

Existe una gran confusión entre una red de computadoras y un **sistema distribuido.** La diferencia principal radica en que, en un sistema distribuido, un conjunto de computadoras independientes aparece ante sus usuarios como un sistema consistente y único. Por lo general, tiene un modelo o paradigma único que se presenta a los usuarios. Con frecuencia, una capa de software que se ejecuta sobre el sistema operativo, denominada **middleware,** es la responsable de implementar este modelo. Un ejemplo bien conocido de un sistema distribuido es **World Wide Web,** en la cual todo se ve como un documento (una página Web).

En una red de computadoras no existe esta consistencia, modelo ni software. Los usuarios están expuestos a las máquinas reales, y el sistema no hace ningún intento porque las máquinas se vean y actúen de manera similar. Si las máquinas tienen hardware diferente y distintos sistemas operativos, eso es completamente transparente para los usuarios. Si un usuario desea ejecutar un programa de una máquina remota, debe registrarse en ella y ejecutarlo desde ahí.

De hecho, un sistema distribuido es un sistema de software construido sobre una red. El software le da un alto grado de consistencia y transparencia. De este modo, la diferencia entre una red y un sistema distribuido está en el software (sobre todo en el sistema operativo), más que en el hardware.

No obstante, tienen muchas cosas en común. Por ejemplo, tanto los sistemas distribuidos como las redes de computadoras necesitan mover archivos. La diferencia está en quién invoca el movimiento, el sistema o el usuario. Aunque el objetivo principal de este libro son las redes, muchos de los temas se relacionan con los sistemas distribuidos. Para más información acerca de los sistemas distribuidos, vea (Tanenbaum y Van Steen, 2002).

1.1 USOS DE LAS REDES DE COMPUTADORAS

Antes de empezar a examinar con detalle los elementos técnicos, vale la pena dedicar algo de tiempo a precisar por qué la gente se interesa en las redes de computadoras y para qué se pueden utilizar. Después de todo, si nadie se hubiera interesado en ellas, no se habrían construido tantas. Empezaremos con el uso tradicional que les dan las empresas y los individuos, y luego avanzaremos a los últimos desarrollos con respecto a los usuarios móviles y la conexión de redes domésticas.

1.1.1 Aplicaciones de negocios

Muchas compañías tienen una cantidad considerable de computadoras. Por ejemplo, una compañía podría tener computadoras separadas para supervisar la producción, controlar inventarios y hacer la nómina. Al principio estas computadoras tal vez hayan trabajado por separado pero, en algún momento, la administración decidió conectarlas para extraer y correlacionar información acerca de toda la compañía.

Dicho de una manera más general, el asunto aquí es la **compartición de recursos** y el objetivo es hacer que todos los programas, el equipo y, en particular, los datos estén disponibles para todos los que se conecten a la red, independientemente de la ubicación física del recurso y del usuario. Un ejemplo claro y muy difundido es el de un grupo de oficinistas que comparten una impresora. Ninguno de los individuos necesita una impresora privada, y una impresora de alto volumen en red suele ser más barata, rápida y fácil de mantener que varias impresoras individuales.

Sin embargo, compartir información es tal vez más importante que compartir recursos físicos, como impresoras, escáneres y quemadores de CDs. Para las compañías grandes y medianas, así como para muchas pequeñas, la información computarizada es vital. La mayoría de las compañías tiene en línea registros de clientes, inventarios, cuentas por cobrar, estados financieros, información de impuestos, etcétera. Si todas las computadoras de un banco se cayeran, éste no duraría más de cinco minutos. Una moderna planta manufacturera, con una línea de ensamblado controlada por computadora, ni siquiera duraría ese tiempo. Incluso una pequeña agencia de viajes o un despacho jurídico de tres personas, ahora dependen en gran medida de las redes de computadoras para que sus empleados puedan tener acceso de manera instantánea a la información y a los documentos importantes.

En las compañías más pequeñas, es posible que todas las computadoras estén en una sola oficina o en un solo edificio, pero en las más grandes, las computadoras y los empleados pueden estar dispersos en docenas de oficinas y plantas en varios países. No obstante, un vendedor en Nueva York podría requerir algunas veces tener acceso a una base de datos de inventario de productos que se encuentra en Singapur. En otras palabras, el hecho de que un usuario esté a 15,000 km de sus datos no debe ser impedimento para que utilice esos datos como si fueran locales. Esta meta se podría resumir diciendo que es un intento por acabar con la “tiranía de la geografía”.

En términos aún más sencillos, es posible imaginar el sistema de información de una compañía como si consistiera en una o más bases de datos y algunos empleados que necesitan acceder a

ellas de manera remota. En este modelo, los datos están almacenados en computadoras poderosas que se llaman **servidores.** Con frecuencia, éstos se encuentran alojados en una central y un administrador de sistemas les da mantenimiento. En contraste, los empleados tienen en sus escritorios máquinas más sencillas, llamadas **clientes,** con las que pueden acceder a datos remotos —por ejemplo, para incluirlos en las hojas de cálculo que están elaborando. (Algunas veces nos referiremos a los usuarios de las máquinas como "el cliente", pero debe quedar claro, por el contexto, si el término se refiere a la computadora o a su usuario.) Las máquinas cliente y servidor están conectadas por una red, como se ilustra en la figura 1-1. Observe que hemos representado a la red como un óvalo sencillo, sin detalle alguno. Utilizaremos esta forma cuando nos refiramos a una red en sentido general. Cuando se requieran más detalles, los proporcionaremos.

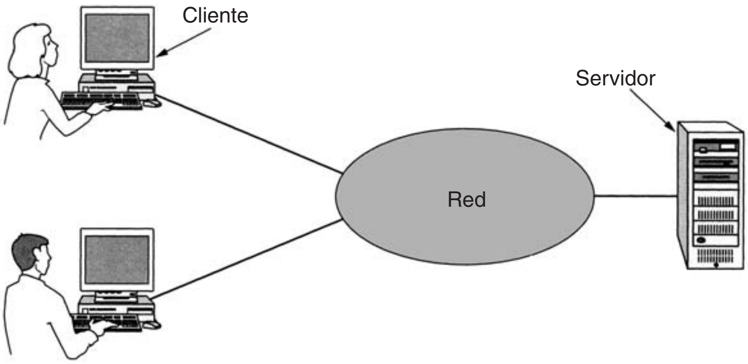


Figura 1-1. Una red con dos clientes y un servidor.

Este conjunto se conoce como **modelo cliente-servidor.** Se utiliza ampliamente y forma la base en gran medida del uso de redes. Es aplicable cuando el cliente y el servidor están en el mismo edificio (por ejemplo, cuando pertenecen a la misma compañía), pero también cuando están bastante retirados. Por ejemplo, cuando una persona en casa accede a una página Web, se emplea el mismo modelo, en el que el servidor remoto de Web es el servidor y la computadora personal del usuario es el cliente. En la mayoría de los casos, un servidor puede manejar una gran cantidad de clientes.

Si vemos el modelo cliente-servidor en detalle, nos daremos cuenta de que hay dos procesos involucrados, uno en la máquina cliente y otro en la máquina servidor. La comunicación toma la siguiente forma: el proceso cliente envía una solicitud a través de la red al proceso servidor y espera una respuesta. Cuando el proceso servidor recibe la solicitud, realiza el trabajo que se le pide o busca los datos solicitados y devuelve una respuesta. Estos mensajes se muestran en la figura 1-2.

Un segundo objetivo de la configuración de una red de computadoras tiene que ver más con la gente que con la información e, incluso, con las computadoras mismas. Una red de computadoras

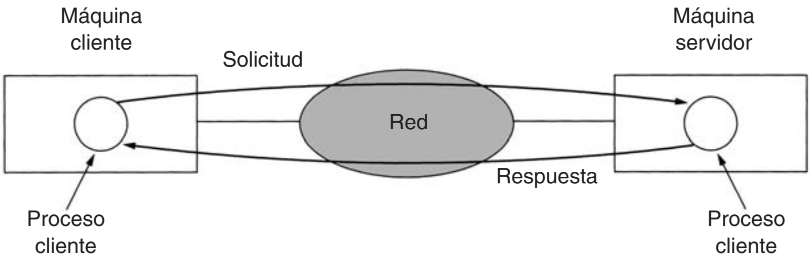


Figura 1-2. El modelo cliente-servidor implica solicitudes y respuestas.

es un poderoso **medio de comunicación** entre los empleados. Casi todas las compañías que tienen dos o más computadoras cuentan con **correo electrónico,** mediante el cual los empleados mantienen generalmente una comunicación diaria. De hecho, una queja común es la gran cantidad de correo electrónico que tenemos que atender, mucho de él sin sentido porque los jefes han descubierto que pueden enviar el mismo mensaje (a menudo sin contenido) a todos sus subordinados con sólo oprimir un botón.

Pero el correo electrónico no es la única forma de comunicación mejorada que las redes de computadoras hacen posible. Con una red es fácil que dos o más personas que trabajan a distancia escriban en conjunto un informe. Si un empleado hace un cambio a un documento en línea, los demás pueden ver el cambio de inmediato, en vez de esperar una carta durante varios días. Esta agilización facilita la cooperación entre grupos de personas que no se encuentran en el mismo lugar, lo cual antes había sido imposible.

Otra forma de comunicación asistida por computadora es la videoconferencia. Con esta tecnología, los empleados en ubicaciones distantes pueden tener una reunión, viéndose y escuchándose unos a otros e incluso escribiendo en una pizarra virtual compartida. La videoconferencia es una herramienta poderosa para eliminar el costo y el tiempo que anteriormente se empleaba en viajar. A veces se dice que la comunicación y el transporte están en competencia, y que el que gane hará obsoleto al otro.

Una tercera meta para cada vez más compañías es hacer negocios de manera electrónica con otras compañías, sobre todo proveedores y clientes. Por ejemplo, los fabricantes de automóviles, de aviones, de computadoras, etcétera, compran subsistemas de diversos proveedores y luego ensamblan las partes. Mediante las redes de computadoras los fabricantes pueden hacer pedidos electrónicamente conforme se requieran. Tener la capacidad de hacer pedidos en tiempo real (es decir, conforme se requieren) reduce la necesidad de tener grandes inventarios y mejora la eficiencia.

Una cuarta meta que se está volviendo más importante es la de hacer negocios con consumidores a través de Internet. Las líneas aéreas, las librerías y los vendedores de música han descubierto que muchos consumidores prefieren realizar sus compras desde casa. Por consiguiente, muchas compañías proporcionan en línea catálogos de sus productos y servicios y levantan pedidos de la misma manera. Se espera que este sector crezca rápidamente en el futuro. Es lo que se conoce como **comercio electrónico.**

1.1.2 Aplicaciones domésticas

En 1977 Ken Olsen era presidente de Digital Equipment Corporation, que en esa época era el segundo proveedor de computadoras en el mundo (después de IBM). Cuando se le preguntó por qué Digital no perseguía el mercado de las computadoras personales en gran volumen, contestó: "No hay razón alguna para que un individuo tenga una computadora en su casa". La historia demostró lo contrario y Digital ya no existe. ¿Por qué la gente compra computadoras para uso doméstico? En principio, para procesamiento de texto y juegos, pero en los últimos años esto ha cambiado radicalmente. Tal vez la razón más importante ahora sea por el acceso a Internet. Algunos de los usos más comunes de Internet por parte de usuarios domésticos son los siguientes:

1. Acceso a información remota.
2. Comunicación de persona a persona.
3. Entretenimiento interactivo.
4. Comercio electrónico.

El acceso a la información remota se puede realizar por diversas razones. Puede ser que navegue por World Wide Web para obtener información o sólo por diversión. La información disponible incluye artes, negocios, cocina, gobiernos, salud, historia, pasatiempos, recreación, ciencia, deportes, viajes y muchas otras cosas más. La diversión viene en demasiadas formas como para mencionarlas, más algunas otras que es mejor no mencionar.

Muchos periódicos ahora están disponibles en línea y pueden personalizarse. Por ejemplo, en algunos casos le puede indicar a un periódico que desea toda la información acerca de políticos corruptos, incendios, escándalos que involucran a las celebridades y epidemias, pero nada sobre fútbol. Incluso puede hacer que los artículos que usted desea se descarguen en su disco duro o se impriman mientras usted duerme, para que cuando se levante a desayunar los tenga disponibles. Mientras continúe esta tendencia, se provocará el desempleo masivo de los niños de 12 años que entregan los diarios, pero los periódicos lo quieren así porque la distribución siempre ha sido el punto débil en la gran cadena de producción.

El tema más importante después de los periódicos (además de las revistas y periódicos científicos) son las bibliotecas digitales en línea. Muchas organizaciones profesionales, como la ACM (*www.acm.org*) y la Sociedad de Computación del IEEE (*www.computer.org*), ya cuentan con muchos periódicos y presentaciones de conferencias en línea. Otros grupos están creciendo de manera rápida. Dependiendo del costo, tamaño y peso de las computadoras portátiles, los libros impresos podrían llegar a ser obsoletos. Los escépticos deben tomar en cuenta el efecto que la imprenta tuvo sobre los manuscritos ilustrados del medioevo.

Todas las aplicaciones anteriores implican las interacciones entre una persona y una base de datos remota llena de información. La segunda gran categoría del uso de redes es la comunicación de persona a persona, básicamente la respuesta del siglo XXI al teléfono del siglo XIX. Millones de personas en todo el mundo utilizan a diario el correo electrónico y su uso está creciendo rápidamente. Ya es muy común que contenga audio y vídeo, así como texto y figuras. Los aromas podrían tardar un poco más.

Muchas personas utilizan los **mensajes instantáneos.** Esta característica, derivada del programa *talk* de UNIX, que se utiliza aproximadamente desde 1970, permite que las personas se escriban mensajes en tiempo real. Una versión, para varias personas, de esta idea es el **salón de conversación** (*chat room*), en el que un grupo de personas puede escribir mensajes para que todos los vean.

Los grupos de noticias mundiales, con debates sobre todo tema imaginable, ya son un lugar común entre un grupo selecto de personas y este fenómeno crecerá para abarcar a la población en general. Estos debates, en los que una persona envía un mensaje y todos los demás suscriptores del grupo de noticias lo pueden leer, van desde los humorísticos hasta los apasionados. A diferencia de los salones de conversación, los grupos de noticias no son en tiempo real y los mensajes se guardan para que cuando alguien vuelva de vacaciones, encuentre todos los mensajes que hayan sido enviados en el ínterin, esperando pacientemente a ser leídos.

Otro tipo de comunicación de persona a persona a menudo se conoce como comunicación de **igual a igual** (*peer to peer*), para distinguirla del modelo cliente-servidor (Parameswaran y cols., 2001). De esta forma, los individuos que forman un grupo esparcido se pueden comunicar con otros del grupo, como se muestra en la figura 1-3. Cada persona puede, en principio, comunicarse con una o más personas; no hay una división fija de clientes y servidores.

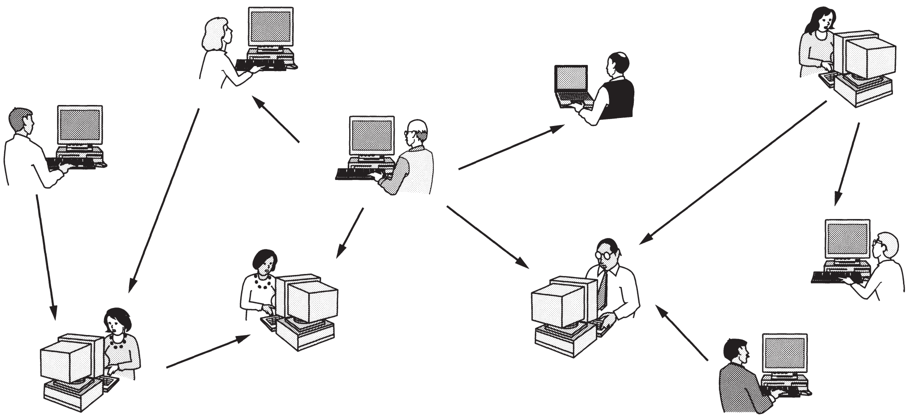


Figura 1-3. En el sistema de igual a igual no hay clientes ni servidores fijos.

La comunicación de igual a igual dominó la mayor parte del 2000 con un servicio llamado Napster, que en su mayor apogeo tenía más de 50 millones de personas canjeando música, lo que fue probablemente la mayor infracción a derechos de autor en toda la historia de la música grabada (Lam y Tan, 2001, y Macedonia, 2000). La idea era muy sencilla. Los miembros registraban en una base de datos central mantenida en el servidor de Napster la música que tenían en sus discos duros. Si un miembro deseaba una canción, verificaba la base de datos para ver quién la tenía e iba directamente ahí para obtenerla. Al no conservar realmente ninguna obra musical en las máquinas, Napster argumentaba que no estaba infringiendo los derechos de autor de nadie. Las cortes no estuvieron de acuerdo y lo clausuraron.

Sin embargo, la siguiente generación de sistemas de igual a igual elimina la base de datos central al hacer que cada usuario mantenga su propia base de datos de manera local, y al proporcionarle una lista de otras personas cercanas que también son miembros del sistema. De esta manera, un nuevo usuario puede ir a cualquiera de ellas para ver qué tiene y obtener una lista de otras más para indagar acerca de más música y más nombres. Este proceso de consulta se puede repetir de manera indefinida hasta construir una enorme base de datos local de lo que hay a disposición. Es una actividad que podría ser tediosa para las personas pero que para las computadoras es muy sencilla.

También existen las aplicaciones legales para la comunicación de igual a igual. Por ejemplo, un club de admiradores que comparte un dominio público de música o cintas de muestra que las nuevas bandas han cedido para efectos de publicidad, familias que comparten fotografías, películas e información genealógica y adolescentes que juegan en línea juegos para varias personas. De hecho, una de las aplicaciones de Internet más populares, el correo electrónico, es esencialmente de igual a igual. Se espera que esta forma de comunicación crezca con rapidez en el futuro.

Los delitos electrónicos no se limitan a la ley de derechos de autor. Otra área activa es la de los juegos electrónicos. Las computadoras han simulado cosas durante décadas. ¿Por qué no simular máquinas tragamonedas, ruedas de la fortuna, repartidores de blackjack y más equipo de juegos electrónicos? El problema es que los juegos electrónicos son legales en muchos lugares (Inglaterra, por ejemplo) y los propietarios de casinos han aprovechado el potencial de los juegos electrónicos por Internet. ¿Qué pasaría si el jugador y el casino estuvieran en países diferentes entre los cuales hay conflicto de leyes? Ésa es una buena pregunta.

Otras aplicaciones orientadas a la comunicación y de rápido crecimiento incluyen el uso de Internet para transportar llamadas telefónicas, el teléfono con vídeo y la radio por Internet. Otra aplicación es el teleaprendizaje, es decir, asistir a clases a las 8:00 A.M. sin el inconveniente de tener que levantarse antes de la cama. A largo plazo, el uso de las redes para mejorar la comunicación de persona a persona puede demostrar que ésta es el área más importante.

Nuestra tercera categoría es el entretenimiento, que es una industria grande y en crecimiento. La aplicación dominante (la que podría impulsar al resto) es el vídeo bajo demanda. De aquí a 10 años, podría seleccionar cualquier película o programa de televisión producido en cualquier país y proyectarlo en su pantalla al instante. Las películas nuevas podrían llegar a ser interactivas, en las que se pediría ocasionalmente al usuario que eligiera el rumbo de la narración, con escenarios alternativos preparados para todos los casos. La televisión en vivo también podría llegar a ser interactiva, permitiendo que la audiencia participe en programas de preguntas, elija entre los competidores, etcétera.

Por otra parte, tal vez el vídeo bajo demanda no sea la aplicación dominante. Podría ser la de los juegos. En la actualidad ya contamos con juegos de simulación de varias personas en tiempo real, como el de las escondidas en un calabozo virtual y simuladores de vuelo en los que los jugadores de un equipo tratan de derribar a los del equipo contrario. Si los juegos se juegan con anteojos y tiempo real tridimensional, con imágenes en movimiento de calidad fotográfica, tenemos un tipo de realidad virtual compartida a nivel mundial.

Nuestra cuarta categoría es el comercio electrónico en el más amplio sentido de la palabra. Comprar desde el hogar ya es una actividad común y permite que los usuarios inspeccionen los

catálogos en línea de miles de compañías. Algunos de estos catálogos proporcionarán pronto la capacidad de obtener un vídeo instantáneo de cualquier producto con sólo hacer clic en el nombre de éste. Si un cliente compra un producto por vía electrónica y no sabe cómo usarlo, podrá consultar el soporte técnico en línea.

Otra área en la que el comercio electrónico ya se está dando es en las instituciones financieras. Mucha gente ya efectúa sus pagos, administra sus cuentas bancarias y maneja sus inversiones de manera electrónica. Seguramente esto crecerá en cuanto las redes sean más seguras.

Un área que prácticamente nadie previó son los mercados de pulgas electrónicos. Las subastas en línea de artículos de segunda mano se han convertido en una industria masiva. A diferencia del comercio electrónico tradicional, que sigue el modelo cliente-servidor, las subastas en línea son más que un sistema de igual a igual, un tipo de sistema de consumidor a consumidor. Algunas de estas formas de comercio electrónico han adoptado una serie de etiquetas con base en que "to" y "2" (en inglés) suenan igual. La figura 1-4 presenta una lista de las abreviaturas más comunes.

Etiqueta	Nombre completo	Ejemplo
B2C	Negocio a consumidor	Pedido de libros en línea
B2B	Negocio a negocio	La fábrica de automóviles hace un pedido de llantas al proveedor
G2C	Gobierno a consumidor	El gobierno distribuye formas fiscales electrónicamente
C2C	Consumidor a consumidor	Subasta en línea de productos de segunda mano
P2P	Igual a igual	Compartición de archivos

Figura 1-4. Algunas formas de comercio electrónico.

Sin duda, el rango de usos de las redes de computadoras crecerá con rapidez y probablemente en formas que nadie puede prever ahora. Después de todo, ¿cuánta gente pudo predecir en 1990 que en diez años las personas podrían escribir mensajes breves en teléfonos celulares durante sus viajes en autobús, lo cual podría ser una forma muy ventajosa para que las compañías telefónicas ganaran dinero? Sin embargo, en la actualidad el servicio de mensajes breves es muy rentable.

Las redes de computadoras podrían llegar a ser sumamente importantes para la gente que no vive en las grandes ciudades, pues les da el mismo acceso a servicios que a las personas que sí viven en ellas. El teleaprendizaje podría afectar radicalmente la educación; las universidades podrían dar servicio a estudiantes nacionales o internacionales. La telemedicina está en inicio (por ejemplo, se utiliza para la supervisión remota de un paciente), pero puede llegar a ser muy importante. Sin embargo, la aplicación clave podría ser algo mundano, como utilizar una *webcam* (cámara conectada a Internet) en su refrigerador, para saber si tiene que comprar leche al regresar del trabajo.

1.1.3 Usuarios móviles

Las computadoras portátiles, como las *notebook* y los asistentes personales digitales (PDAs), son uno de los segmentos de crecimiento más rápido de la industria de la computación. Muchos propietarios de estas computadoras poseen máquinas de escritorio en la oficina y desean estar conectados a su base doméstica cuando están de viaje o fuera de casa. Puesto que no

es posible tener una conexión alámbrica en autos y aviones, hay un gran interés en las redes inalámbricas. En esta sección veremos brevemente algunos usos de ellas.

¿Por qué querría alguien una? Un argumento común es la oficina portátil. Con frecuencia, las personas que están de viaje desean utilizar sus equipos portátiles para enviar y recibir llamadas telefónicas, faxes y correo electrónico, navegar en Web, acceder a archivos remotos e iniciar sesión en máquinas remotas. Y desean hacer esto desde cualquier punto, ya sea por tierra, mar o aire. Por ejemplo, actualmente en las conferencias por computadora, los organizadores suelen configurar una red inalámbrica en el área de la conferencia. Cualquiera que tenga una computadora portátil y un módem inalámbrico puede conectarse a Internet, como si la computadora estuviera conectada a una red alámbrica (cableada). Del mismo modo, algunas universidades han instalado redes inalámbricas en sus campus para que los estudiantes se puedan sentar entre los árboles y consultar los archivos de la biblioteca o leer su correo electrónico.

Las redes inalámbricas son de gran utilidad para las flotas de camiones, taxis, vehículos de entrega y reparadores, para mantenerse en contacto con la casa. Por ejemplo, en muchas ciudades los taxistas trabajan por su cuenta, más que para una empresa de taxis. En algunas de estas ciudades, los taxis tienen una pantalla que el conductor puede ver. Cuando el cliente solicita un servicio, un despachador central escribe los puntos en los que el chofer deberá recoger y dejar al cliente. Esta información se despliega en las pantallas de los conductores y suena un timbre. El conductor que oprima primero un botón en la pantalla recibe la llamada.

Las redes inalámbricas también son importantes para la milicia. Si tiene que estar disponible en breve para pelear una guerra en cualquier parte de la Tierra, probablemente no sea bueno pensar en utilizar la infraestructura de conectividad de redes local. Lo mejor sería tener la propia.

Aunque la conectividad inalámbrica y la computación portátil se relacionan frecuentemente, no son idénticas, como se muestra en la figura 1-5, en la que vemos una diferencia entre **inalámbrica fija** e **inalámbrica móvil.** Incluso en ocasiones las computadoras portátiles son alámbricas. Por ejemplo, si un viajero conecta una portátil a una toma telefónica en su habitación del hotel, tiene movilidad sin una red inalámbrica.

Inalámbrica	Móvil	Aplicaciones
No	No	Computadoras de escritorio en oficinas
No	Sí	Una computadora portátil usada en un cuarto de hotel
Sí	No	Redes en construcciones antiguas sin cableado
Sí	Sí	Oficina portátil; PDA para inventario de almacén

Figura 1-5. Combinaciones de redes inalámbricas y computación móvil.

Por otra parte, algunas computadoras inalámbricas no son móviles. Un ejemplo representativo sería una compañía que posee un edificio antiguo que no tiene cableado de redes y que desea conectar sus computadoras. La instalación de una red inalámbrica podría requerir un poco más que comprar una caja pequeña con algunos aparatos electrónicos, desempacarlos y conectarlos. Sin embargo, esta solución podría ser mucho más barata que contratar trabajadores que coloquen ductos de cable para acondicionar el edificio.

Desde luego, también existen las aplicaciones inalámbricas móviles, que van desde la oficina portátil hasta las personas que pasean por una tienda con un PDA realizando un inventario. En muchos aeropuertos, los empleados de alquiler de coches trabajan en los estacionamientos con computadoras portátiles inalámbricas. Escriben el número de la placa de circulación de los autos alquilados, y su computadora portátil, que tiene una impresora integrada, llama a la computadora principal, obtiene la información del arrendamiento e imprime la factura en el acto.

Conforme se extienda la tecnología inalámbrica, es probable que surjan otras aplicaciones. Echemos un vistazo a algunas de las posibilidades. Los parquímetros inalámbricos tienen ventajas para los usuarios y las autoridades administrativas gubernamentales. Los medidores pueden aceptar tarjetas de crédito o de débito y verificarlas de manera instantánea a través del vínculo inalámbrico. Cuando un medidor expire, se podría verificar la presencia de un auto (emitiendo una señal) y reportar la expiración a la policía. Se ha estimado que con esta medida, los gobiernos de las ciudades de Estados Unidos podrían colectar $10 mil millones adicionales (Harte y cols., 2000). Además, la entrada en vigor del aparcamiento ayudaría al ambiente, debido a que los conductores que al saber que podrían ser detenidos al estacionarse de manera ilegal, utilizarían el transporte público.

Los expendedores automáticos de alimentos, bebidas, etcétera, se encuentran por todas partes. Sin embargo, los alimentos no entran en las máquinas por arte de magia. Periódicamente, alguien va con un camión y las llena. Si los expendedores automáticos emitieran informes periódicos una vez al día en los que indicaran sus inventarios actuales, el conductor del camión sabría qué máquinas necesitan servicio y qué cantidad de qué productos llevar. Esta información podría conducir a una mayor eficiencia en la planeación de las rutas. Desde luego que esta información también se podría enviar a través de un teléfono de línea común, pero proporcionar a cada expendedor automático una conexión fija telefónica para que realice una llamada al día es costoso debido a los cargos fijos mensuales.

Otra área en la que la tecnología inalámbrica podría ahorrar dinero es en la lectura de medidores de servicios públicos. Si los medidores de electricidad, gas, agua y otros servicios domésticos reportaran su uso a través de una red inalámbrica, no habría necesidad de enviar lectores de medidores. Del mismo modo, los detectores inalámbricos de humo podrían comunicarse con el departamento de bomberos en lugar de hacer tanto ruido (lo cual no sirve de nada si no hay nadie en casa). Conforme baje el costo de los dispositivos de radio y el tiempo aire, más y más medidas e informes se harán a través de redes inalámbricas.

Un área de aplicación totalmente diferente para las redes inalámbricas es la fusión esperada de teléfonos celulares y PDAs en computadoras inalámbricas diminutas. Un primer intento fue el de los diminutos PDAs que podían desplegar páginas Web reducidas al mínimo en sus pequeñas pantallas. Este sistema, llamado **WAP 1.0** (**Protocolo de Aplicaciones Inalámbricas**)**,** falló en gran parte debido a sus pantallas microscópicas, bajo ancho de banda y servicio deficiente. Pero con WAP 2.0 serán mejores los dispositivos y servicios nuevos.

La fuerza que impulsa estos dispositivos es la llamada **comercio móvil** (*m-commerce*) (Senn, 2000). La fuerza que impulsa este fenómeno consiste en diversos fabricantes de PDAs inalámbricos y operadores de redes que luchan por descubrir cómo ganar una parte del pastel del comercio móvil. Una de sus esperanzas es utilizar los PDAs inalámbricos para servicios bancarios y de compras. Una idea es utilizar los PDAs inalámbricos como un tipo de cartera electrónica, que

autorice pagos en tiendas como un reemplazo del efectivo y las tarjetas de crédito. De este modo, el cargo aparecerá en la factura del teléfono celular. Desde el punto de vista de la tienda, este esquema le podría ahorrar la mayor parte de la cuota de la empresa de tarjetas de crédito, que puede ser un porcentaje importante. Desde luego, este plan puede resultar contraproducente, puesto que los clientes que están en una tienda podrían utilizar los PDAs para verificar los precios de la competencia antes de comprar. Peor aún, las compañías telefónicas podrían ofrecer PDAs con lectores de códigos de barras que permitan a un cliente rastrear un producto en una tienda y obtener en forma instantánea un informe detallado de dónde más se puede comprar y a qué precio.

Puesto que el operador de redes sabe dónde está el usuario, algunos servicios se hacen intencionalmente dependientes de la ubicación. Por ejemplo, se podría preguntar por una librería cercana o un restaurante chino. Los mapas móviles y los pronósticos meteorológicos muy locales ("¿Cuándo va a dejar de llover en mi traspatio?") son otros candidatos. Sin duda, aparecerán otras muchas aplicaciones en cuanto estos dispositivos se difundan más ampliamente.

Un punto muy importante para el comercio móvil es que los usuarios de teléfonos celulares están acostumbrados a pagar por todo (en contraste con los usuarios de Internet, que esperan recibir prácticamente todo sin costo). Si un sitio Web cobrara una cuota por permitir a sus clientes pagar con tarjeta de crédito, provocaría una reclamación muy ruidosa de los usuarios. Si un operador de telefonía celular permitiera que las personas pagaran artículos en una tienda utilizando el teléfono celular y luego cargara una cuota por este servicio, probablemente sus clientes lo aceptarían como algo normal. Sólo el tiempo lo dirá.

Un poco más lejanas están las redes de área personal y las microcomputadoras personales de bolsillo. IBM ha desarrollado un reloj que ejecuta Linux (el cual incluye el sistema de ventanas X11) y tiene conectividad inalámbrica a Internet para enviar y recibir correo electrónico (Narayanaswami y cols., 2002). En el futuro, las personas podrían intercambiar tarjetas de presentación con sólo exponer sus relojes entre sí. Las computadoras de bolsillo inalámbricas pueden dar acceso a las personas a sitios seguros de la misma manera en que lo hacen las tarjetas de banda magnética (posiblemente en combinación con un código de PIN o medición biométrica). Estos relojes también podrían recuperar información relativa a la ubicación actual del usuario (por ejemplo, restaurantes locales). Las posibilidades son infinitas.

Los relojes inteligentes con radio han sido parte de nuestro espacio mental desde que aparecieron en las tiras cómicas de Dick Tracy, en 1946. Pero, ¿polvo inteligente? Los investigadores en Berkeley han empaquetado una computadora inalámbrica en un cubo de 1 mm por lado (Warneke y cols., 2001). Entre las aplicaciones potenciales se incluyen el seguimiento de inventarios, paquetes e incluso pequeños pájaros, roedores e insectos.

1.1.4 Temas sociales

La amplia introducción de las redes ha presentado problemas sociales, éticos y políticos. Mencionemos brevemente algunos de ellos; un estudio completo requeriría todo un libro, por lo menos. Un rasgo popular de muchas redes son los grupos de noticias o boletines electrónicos mediante los cuales las personas pueden intercambiar mensajes con individuos de los mismos intereses. Siempre y cuando los asuntos se restrinjan a temas técnicos o pasatiempos como la jardinería, no surgirán demasiados problemas.

El problema viene cuando los grupos de noticias se enfocan en temas que las personas en realidad tocan con cuidado, como política, religión o sexo. Los puntos de vista enviados a tales grupos podrían ser ofensivos para algunas personas. Peor aún, podrían no ser políticamente correctos. Además, los mensajes no tienen que limitarse a texto. En la actualidad se pueden enviar fotografías en alta resolución e incluso pequeños videoclips a través de redes de computadoras. Algunas personas practican la filosofía de vive y deja vivir, pero otras sienten que enviar cierto material (por ejemplo, ataques a países o religiones en particular, pornografía, etcétera) es sencillamente inaceptable y debe ser censurado. Los diversos países tienen diferentes y conflictivas leyes al respecto. De esta manera, el debate se aviva.

Las personas han demandado a los operadores de redes, afirmando que son responsables, como sucede en el caso de los periódicos y las revistas, del contenido que transmiten. La respuesta inevitable es que una red es como una compañía de teléfonos o la oficina de correos, por lo que no se puede esperar que vigilen lo que dicen los usuarios. Más aún, si los operadores de redes censuraran los mensajes, borrarían cualquier contenido que contuviera incluso la mínima posibilidad de que se les demandara, pero con esto violarían los derechos de sus usuarios a la libre expresión. Probablemente lo más seguro sería decir que este debate seguirá durante algún tiempo.

Otra área divertida es la de los derechos de los empleados en comparación con los de los empleadores. Muchas personas leen y escriben correo electrónico en el trabajo. Muchos empleadores han exigido el derecho a leer y, posiblemente, censurar los mensajes de los empleados, incluso los enviados desde un equipo doméstico después de las horas de trabajo. No todos los empleados están de acuerdo con esto.

Incluso si los empleadores tienen poder sobre los empleados, ¿esta relación también rige a las universidades y los estudiantes? ¿Qué hay acerca de las escuelas secundarias y los estudiantes? En 1994, la Carnegie-Mellon University decidió suspender el flujo de mensajes entrantes de varios grupos de noticias que trataban sexo porque la universidad sintió que el material era inapropiado para menores (es decir, menores de 18 años). Tomó años recuperarse de este suceso.

Otro tema de importancia es el de los derechos del gobierno y los de los ciudadanos. El FBI ha instalado un sistema en muchos proveedores de servicios de Internet para curiosear entre todos los correos electrónicos en busca de fragmentos que le interesen (Blaze y Bellovin, 2000; Sobel, 2001; Zacks, 2001). El sistema se llamaba originalmente **Carnivore** pero la mala publicidad provocó que se cambiara el nombre por uno menos agresivo que sonara como DCS1000. Pero su objetivo sigue siendo el de espiar a millones de personas con la esperanza de encontrar información acerca de actividades ilegales. Por desgracia, la Cuarta Enmienda de la Constitución de Estados Unidos prohíbe que el gobierno realice investigaciones sin una orden de cateo. Decidir si estas palabras, escritas en el siglo XVIII, aún son válidas en el siglo XXI es un asunto que podría mantener ocupadas a las cortes hasta el siglo XXII.

El gobierno no tiene el monopolio de las amenazas contra la privacidad de una persona. El sector privado también hace su parte. Por ejemplo, los archivos pequeños llamados cookies que los navegadores Web almacenan en las computadoras de los usuarios permiten que las empresas rastreen las actividades de éstos en el ciberespacio, y podrían permitir que los números de tarjeta de crédito, del seguro social y otra información confidencial se divulguen por toda la Internet (Berghel, 2001).

Las redes de computadoras ofrecen la posibilidad de enviar mensajes anónimos. En algunas situaciones esta capacidad podría ser deseable. Por ejemplo, los estudiantes, soldados, empleados y ciudadanos pueden denunciar el comportamiento ilegal de algunos profesores, oficiales, superiores y políticos sin temor a represalias. Por otra parte, en Estados Unidos, y en la mayoría de las democracias, la ley otorga específicamente a una persona acusada el derecho de poder confrontar y desafiar a su acusador en la corte. Las acusaciones anónimas no se pueden usar como evidencia.

En resumen, las redes de computadoras, como la imprenta hace 500 años, permiten que el ciudadano común distribuya sus puntos de vista en diversos modos y a audiencias diferentes, lo cual antes no era posible. Este nuevo fondo de libertad ofrece consigo muchos temas sociales, políticos y morales sin resolver.

Junto con lo bueno viene lo malo. Así parece ser la vida. Internet hace posible encontrar con rapidez información, pero una gran cantidad de ella está mal documentada, es falsa o completamente errónea. El consejo médico que obtuvo en Internet podría haber venido de un ganador del Premio Nobel o de un desertor de la preparatoria. Las redes de computadoras también han introducido nuevos tipos de comportamientos antisociales y criminales. La publicidad no deseada (*spam*) se ha convertido en algo común debido a que algunas personas se dedican a reunir millones de direcciones de correo electrónico y las venden en CD-ROMs a comerciantes. Los mensajes por correo electrónico que contienen elementos activos (básicamente programas o macros que se ejecutan en la máquina del receptor) pueden contener virus potencialmente destructores.

El robo de identidad se ha convertido en un problema grave, ya que los ladrones ahora reúnen información sobre una persona para obtener tarjetas de crédito y otros documentos a nombre de ella. Por último, la capacidad de transmitir música y vídeo de manera digital ha abierto la puerta a violaciones masivas de derechos de autor, que son difíciles de detectar y castigar.

Muchos de estos problemas se podrían resolver si la industria de las computadoras tomara la seguridad de las computadoras con seriedad. Si todos los mensajes se codificaran y autenticaran, sería más difícil que se cometieran delitos. Esta tecnología está bien establecida y la estudiaremos en detalle en el capítulo 8. El problema es que los proveedores de hardware y software saben que poner funciones de seguridad cuesta dinero y que sus clientes no las solicitan. Además, una gran cantidad de los problemas proviene de un software con fallas, debido a que los proveedores saturan de funciones sus programas, lo que implica más código e, inevitablemente, más fallas. Un impuesto a las funciones nuevas podría ayudar, pero eso sería como vender un problema por centavos. Reponer el software defectuoso podría ser bueno, pero eso llevaría a la quiebra a toda la industria del software en el primer año.

1.2 HARDWARE DE REDES

Ya es tiempo de centrar nuevamente la atención en los temas técnicos correspondientes al diseño de redes (la parte de trabajo) y dejar a un lado las aplicaciones y los aspectos sociales de la conectividad (la parte divertida). Por lo general, no hay una sola clasificación aceptada en la que se ajusten todas las redes de computadoras, pero hay dos que destacan de manera importante: la tecnología de transmisión y la escala. Examinaremos cada una a la vez.

En un sentido amplio, hay dos tipos de tecnología de transmisión que se utilizan de manera extensa. Son las siguientes:

1. Enlaces de difusión.
2. Enlaces de punto a punto.

Las **redes de difusión** (*broadcast*) tienen un solo canal de comunicación, por lo que todas las máquinas de la red lo comparten. Si una máquina envía un mensaje corto —en ciertos contextos conocido como **paquete**—, todas las demás lo reciben. Un campo de dirección dentro del paquete especifica el destinatario. Cuando una máquina recibe un paquete, verifica el campo de dirección. Si el paquete va destinado a esa máquina, ésta lo procesa; si va destinado a alguna otra, lo ignora.

En una analogía, imagine a alguien que está parado al final de un corredor con varios cuartos a los lados y que grita: "Jorge, ven. Te necesito". Aunque en realidad el grito (paquete) podría haber sido escuchado (recibido), por muchas personas, sólo Jorge responde (lo procesa). Los demás simplemente lo ignoran. Otra analogía es la de los anuncios en un aeropuerto que piden a todos los pasajeros del vuelo 644 se reporten en la puerta 12 para abordar de inmediato.

Por lo general, los sistemas de difusión también permiten el direccionamiento de un paquete a *todos* los destinos utilizando un código especial en el campo de dirección. Cuando se transmite un paquete con este código, todas las máquinas de la red lo reciben y procesan. Este modo de operación se conoce como **difusión** (*broadcasting*). Algunos sistemas de difusión también soportan la transmisión a un subconjunto de máquinas, algo conocido como **multidifusión** (*multicasting*). Un esquema posible es la reserva de un bit para indicar la multidifusión. Los bits de dirección $n - 1$ restantes pueden contener un número de grupo. Cada máquina puede "suscribirse" a alguno o a todos los grupos. Cuando se envía un paquete a cierto grupo, se distribuye a todas las máquinas que se suscriben a ese grupo.

En contraste, las redes **punto a punto** constan de muchas conexiones entre pares individuales de máquinas. Para ir del origen al destino, un paquete en este tipo de red podría tener que visitar primero una o más máquinas intermedias. A menudo es posible que haya varias rutas o longitudes diferentes, de manera que encontrar las correctas es importante en redes de punto a punto. Por regla general (aunque hay muchas excepciones), las redes más pequeñas localizadas en una misma área geográfica tienden a utilizar la difusión, mientras que las más grandes suelen ser de punto a punto. La transmisión de punto a punto con un emisor y un receptor se conoce como **unidifusión** (*unicasting*).

Un criterio alternativo para la clasificación de las redes es su escala. En la figura 1-6 clasificamos los sistemas de procesadores múltiples por tamaño físico. En la parte superior se muestran las **redes de área personal,** que están destinadas para una sola persona. Por ejemplo, una red inalámbrica que conecta una computadora con su ratón, teclado e impresora, es una red de área personal. Incluso un PDA que controla el audífono o el marcapaso de un usuario encaja en esta categoría. A continuación de las redes de área personal se encuentran redes más grandes. Se pueden dividir en redes de área local, de área metropolitana y de área amplia. Por último, la conexión de dos o más redes se conoce como interred.

Distancia entre procesadores	Procesadores ubicados en el mismo	Ejemplo
1 m	Metro cuadrado	Red de área personal
10 m	Cuarto	
100 m	Edificio	Red de área local
1 km	Campus	
10 km	Ciudad	Red de área metropolitana
100 km	País	Red de área amplia
1,000 km	Continente	
10,000 km	Planeta	Internet

Figura 1-6. Clasificación de procesadores interconectados por escala.

Internet es un ejemplo bien conocido de una interred. La distancia es importante como una clasificación en metros porque se utilizan diferentes técnicas en diferentes escalas. En este libro nos ocuparemos de las redes en todas estas escalas. A continuación se proporciona una breve introducción al hardware de redes.

1.2.1 Redes de área local

Las **redes de área local** (generalmente conocidas como **LANs**) son redes de propiedad privada que se encuentran en un solo edificio o en un campus de pocos kilómetros de longitud. Se utilizan ampliamente para conectar computadoras personales y estaciones de trabajo en oficinas de una empresa y de fábricas para compartir recursos (por ejemplo, impresoras) e intercambiar información. Las LANs son diferentes de otros tipos de redes en tres aspectos: 1) tamaño; 2) tecnología de transmisión, y 3) topología.

Las LANs están restringidas por tamaño, es decir, el tiempo de transmisión en el peor de los casos es limitado y conocido de antemano. El hecho de conocer este límite permite utilizar ciertos tipos de diseño, lo cual no sería posible de otra manera. Esto también simplifica la administración de la red.

Las LANs podrían utilizar una tecnología de transmisión que consiste en un cable al cual están unidas todas las máquinas, como alguna vez lo estuvo parte de las líneas de las compañías telefónicas en áreas rurales. Las LANs tradicionales se ejecutan a una velocidad de 10 a 100 Mbps, tienen un retardo bajo (microsegundos o nanosegundos) y cometen muy pocos errores. Las LANs más nuevas funcionan hasta a 10 Gbps. En este libro continuaremos con lo tradicional y mediremos las velocidades de las líneas en megabits por segundo (1 Mbps es igual a 1,000,000 de bits por segundo) y gigabits por segundo (1 Gbps es igual a 1,000,000,000 de bits por segundo).

Para las LANs de difusión son posibles varias topologías. La figura 1-7 muestra dos de ellas. En una red de bus (es decir, un cable lineal), en cualquier instante al menos una máquina es la maestra y puede transmitir. Todas las demás máquinas se abstienen de enviar. Cuando se presenta el conflicto de que dos o más máquinas desean transmitir al mismo tiempo, se requiere un meca-

nismo de arbitraje. Tal mecanismo podría ser centralizado o distribuido. Por ejemplo, el IEEE 802.3, popularmente conocido como **Ethernet,** es una red de difusión basada en bus con control descentralizado, que por lo general funciona de 10 Mbps a 10 Gbps. Las computadoras que están en una Ethernet pueden transmitir siempre que lo deseen; si dos o más paquetes entran en colisión, cada computadora espera un tiempo aleatorio y lo intenta de nuevo más tarde.

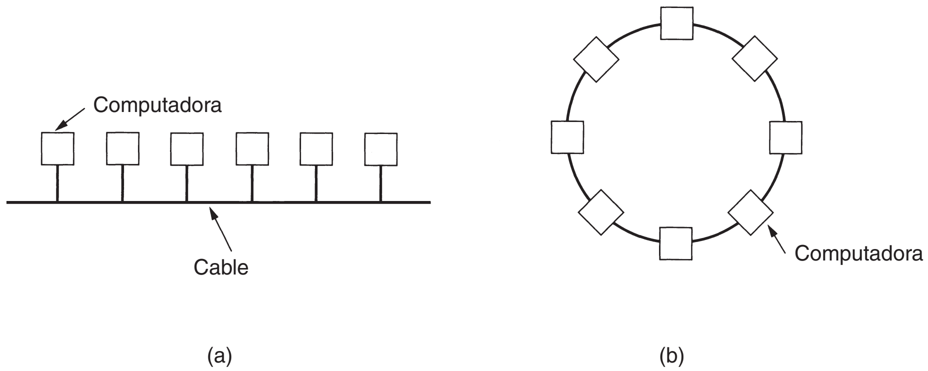


Figura 1-7. Dos redes de difusión. (a) De bus. (b) De anillo.

Un segundo tipo de sistema de difusión es el de anillo. En un anillo, cada bit se propaga por sí mismo, sin esperar al resto del paquete al que pertenece. Por lo común, cada bit navega por todo el anillo en el tiempo que le toma transmitir algunos bits, a veces incluso antes de que se haya transmitido el paquete completo. Al igual que con todos los demás sistemas de difusión, se requieren algunas reglas para controlar los accesos simultáneos al anillo. Se utilizan varios métodos, por ejemplo, el de que las máquinas deben tomar su turno. El IEEE 802.5 (el token ring de IBM) es una LAN basada en anillo que funciona a 4 y 16 Mbps. El FDDI es otro ejemplo de una red de anillo.

Las redes de difusión se pueden dividir aún más en estáticas y dinámicas, dependiendo de cómo se asigne el canal. Una asignación estática típica sería dividir el tiempo en intervalos discretos y utilizar un algoritmo *round-robin*, permitiendo que cada máquina transmita sólo cuando llegue su turno. La asignación estática desperdicia capacidad de canal cuando una máquina no tiene nada que transmitir al llegar su turno, por lo que la mayoría de los sistemas trata de asignar el canal de forma dinámica (es decir, bajo demanda).

Los métodos de asignación dinámica para un canal común pueden ser centralizados o descentralizados. En el método centralizado hay una sola entidad, por ejemplo, una unidad de arbitraje de bus, la cual determina quién sigue. Esto se podría hacer aceptando solicitudes y tomando decisiones de acuerdo con algunos algoritmos internos. En el método descentralizado de asignación de canal no hay una entidad central; cada máquina debe decidir por sí misma cuándo transmitir. Usted podría pensar que esto siempre conduce al caos, pero no es así. Más adelante estudiaremos muchos algoritmos designados para poner orden y evitar el caos potencial.

1.2.2 Redes de área metropolitana

Una **red de área metropolitana** (**MAN**) abarca una ciudad. El ejemplo más conocido de una MAN es la red de televisión por cable disponible en muchas ciudades. Este sistema creció a partir de los primeros sistemas de antena comunitaria en áreas donde la recepción de la televisión al aire era pobre. En dichos sistemas se colocaba una antena grande en la cima de una colina cercana y la señal se canalizaba a las casas de los suscriptores.

Al principio eran sistemas diseñados de manera local con fines específicos. Después las compañías empezaron a pasar a los negocios, y obtuvieron contratos de los gobiernos de las ciudades para cablear toda una ciudad. El siguiente paso fue la programación de televisión e incluso canales designados únicamente para cable. Con frecuencia, éstos emitían programas de un solo tema, como sólo noticias, deportes, cocina, jardinería, etcétera. Sin embargo, desde su inicio y hasta finales de la década de 1990, estaban diseñados únicamente para la recepción de televisión.

A partir de que Internet atrajo una audiencia masiva, los operadores de la red de TV por cable se dieron cuenta de que con algunos cambios al sistema, podrían proporcionar servicio de Internet de dos vías en las partes sin uso del espectro. En ese punto, el sistema de TV por cable empezaba a transformarse de una forma de distribución de televisión a una red de área metropolitana. Para que se dé una idea, una MAN podría verse como el sistema que se muestra en la figura 1-8, donde se aprecia que las señales de TV e Internet se alimentan hacia un **amplificador head end** para enseguida transmitirse a las casas de las personas. En el capítulo 2 trataremos con detalle este tema.

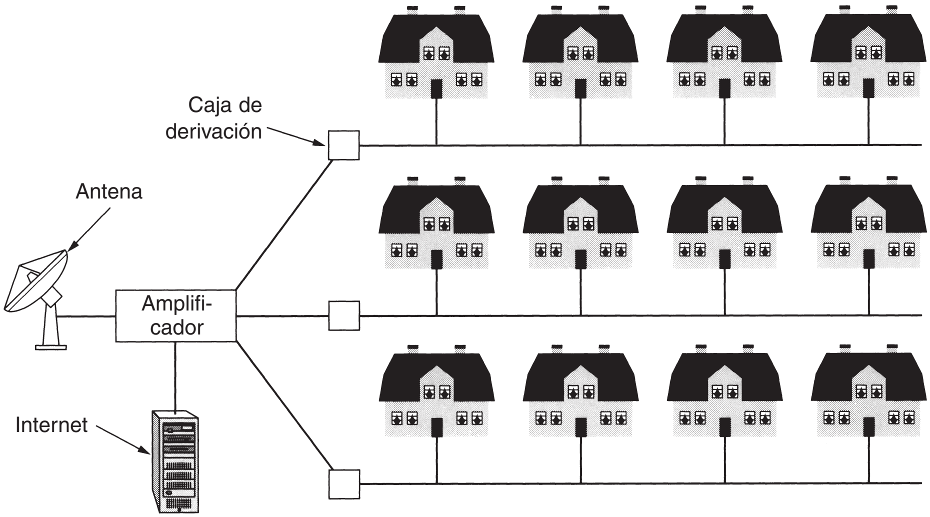


Figura 1-8. Una red de área metropolitana, basada en TV por cable.

La televisión por cable no es solamente una MAN. Desarrollos recientes en el acceso inalámbrico a alta velocidad a Internet dieron como resultado otra MAN, que se estandarizó como IEEE 802.16. En el capítulo 2 veremos esta área.

1.2.3 Redes de área amplia

Una **red de área amplia (WAN),** abarca una gran área geográfica, con frecuencia un país o un continente. Contiene un conjunto de máquinas diseñado para programas (es decir, aplicaciones) de usuario. Seguiremos el uso tradicional y llamaremos **hosts** a estas máquinas. Los *hosts* están conectados por una **subred de comunicación,** o simplemente **subred,** para abreviar. Los clientes son quienes poseen a los *hosts* (es decir, las computadoras personales de los usuarios), mientras que, por lo general, las compañías telefónicas o los proveedores de servicios de Internet poseen y operan la subred de comunicación. La función de una subred es llevar mensajes de un *host* a otro, como lo hace el sistema telefónico con las palabras del que habla al que escucha. La separación de los aspectos de la comunicación pura de la red (la subred) de los aspectos de la aplicación (los *hosts*), simplifica en gran medida todo el diseño de la red.

En la mayoría de las redes de área amplia la subred consta de dos componente distintos: líneas de transmisión y elementos de conmutación. Las **líneas de transmisión** mueven bits entre máquinas. Pueden estar hechas de cable de cobre, fibra óptica o, incluso, radioenlaces. Los **elementos de conmutación** son computadoras especializadas que conectan tres o más líneas de transmisión. Cuando los datos llegan a una línea de entrada, el elemento de conmutación debe elegir una línea de salida en la cual reenviarlos. Estas computadoras de conmutación reciben varios nombres; conmutadores y enrutadores son los más comunes.

En este modelo, que se muestra en la figura 1-9, cada *host* está conectado frecuentemente a una LAN en la que existe un enrutador, aunque en algunos casos un *host* puede estar conectado de manera directa a un enrutador. El conjunto de líneas de comunicación y enrutadores (pero no de *hosts*) forma la subred.

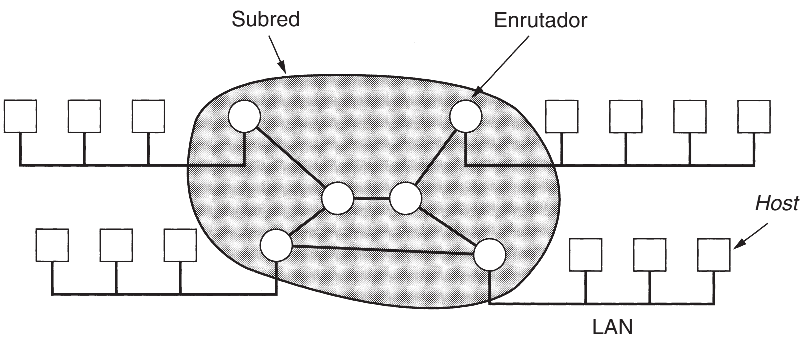


Figura 1-9. Relación entre *hosts* de LANs y la subred.

A continuación se presenta un breve comentario acerca del término “subred”. Originalmente, su **único** significado era el conjunto de enrutadores y líneas de comunicación que movía paquetes del *host* de origen al de destino. Sin embargo, algunos años más tarde también adquirió un segundo

significado junto con el direccionamiento de redes (que expondremos en el capítulo 5). Desgraciadamente, no existe una alternativa de amplio uso con respecto a su significado inicial por lo que, con algunas reservas, utilizaremos este término en ambos sentidos. El contexto dejará en claro su significado.

En la mayoría de las WANs, la red contiene numerosas líneas de transmisión, cada una de las cuales conecta un par de enrutadores. Si dos enrutadores que no comparten una línea de transmisión quieren conectarse, deberán hacerlo de manera indirecta, a través de otros enrutadores. Cuando un paquete es enviado desde un enrutador a otro a través de uno o más enrutadores intermedios, el paquete se recibe en cada enrutador intermedio en su totalidad, se almacena ahí hasta que la línea de salida requerida esté libre y, por último, se reenvía. Una subred organizada a partir de este principio se conoce como subred de **almacenamiento y reenvío** (*store and forward*) o de **conmutación de paquetes.** Casi todas las redes de área amplia (excepto las que utilizan satélites) tienen subredes de almacenamiento y reenvío. Cuando los paquetes son pequeños y tienen el mismo tamaño, se les llama **celdas.**

El principio de una WAN de conmutación de paquetes es tan importante que vale la pena dedicarle algunas palabras más. En general, cuando un proceso de cualquier *host* tiene un mensaje que se va a enviar a un proceso de algún otro *host*, el *host* emisor divide primero el mensaje en paquetes, los cuales tienen un número de secuencia. Estos paquetes se envían entonces por la red de uno en uno en una rápida sucesión. Los paquetes se transportan de forma individual a través de la red y se depositan en el *host* receptor, donde se reensamblan en el mensaje original y se entregan al proceso receptor. En la figura 1-10 se ilustra un flujo de paquetes correspondiente a algún mensaje inicial.

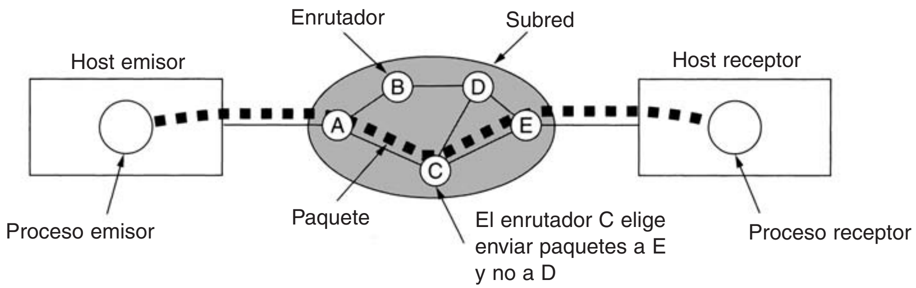


Figura 1-10. Flujo de paquetes desde un emisor a un receptor.

En esta figura todos los paquetes siguen la ruta *ACE* en vez de la *ABDE* o *ACDE*. En algunas redes todos los paquetes de un mensaje determinado *deben* seguir la misma ruta; en otras, cada paquete se enruta por separado. Desde luego, si *ACE* es la mejor ruta, todos los paquetes se podrían enviar a través de ella, incluso si cada paquete se enruta de manera individual.

Las decisiones de enrutamiento se hacen de manera local. Cuando un paquete llega al enrutador *A*, éste debe decidir si el paquete se enviará hacia *B* o hacia *C*. La manera en que el enrutador *A* toma esa decisión se conoce como **algoritmo de enrutamiento.** Existen muchos de ellos. En el capítulo 5 estudiaremos con detalle algunos.

No todas las WANs son de conmutación de paquetes. Una segunda posibilidad para una WAN es un sistema satelital. Cada enrutador tiene una antena a través de la cual puede enviar y recibir. Todos los enrutadores pueden escuchar la salida *desde* el satélite y, en algunos casos, también pueden escuchar las transmisiones de los demás enrutadores *hacia* el satélite. Algunas veces los enrutadores están conectados a una subred de punto a punto elemental, y sólo algunos de ellos tienen una antena de satélite. Por naturaleza, las redes satelital son de difusión y son más útiles cuando la propiedad de difusión es importante.

1.2.4 Redes inalámbricas

La comunicación inalámbrica digital no es una idea nueva. A principios de 1901, el físico italiano Guillermo Marconi demostró un telégrafo inalámbrico desde un barco a tierra utilizando el código Morse (después de todo, los puntos y rayas son binarios). Los sistemas inalámbricos digitales de la actualidad tienen un mejor desempeño, pero la idea básica es la misma.

Como primera aproximación, las redes inalámbricas se pueden dividir en tres categorías principales:

1. Interconexión de sistemas.
2. LANs inalámbricas.
3. WANs inalámbricas.

La interconexión de sistemas se refiere a la interconexión de componentes de una computadora que utiliza radio de corto alcance. La mayoría de las computadoras tiene un monitor, teclado, ratón e impresora, conectados por cables a la unidad central. Son tantos los usuarios nuevos que tienen dificultades para conectar todos los cables en los enchufes correctos (aun cuando suelen estar codificados por colores) que la mayoría de los proveedores de computadoras ofrece la opción de enviar a un técnico a la casa del usuario para que realice esta tarea. En consecuencia, algunas compañías se reunieron para diseñar una red inalámbrica de corto alcance llamada **Bluetooth** para conectar sin cables estos componentes. Bluetooth también permite conectar cámaras digitales, auriculares, escáneres y otros dispositivos a una computadora con el único requisito de que se encuentren dentro del alcance de la red. Sin cables, sin instalación de controladores, simplemente se colocan, se encienden y funcionan. Para muchas personas, esta facilidad de operación es algo grandioso.

En la forma más sencilla, las redes de interconexión de sistemas utilizan el paradigma del maestro y el esclavo de la figura 1-11(a). La unidad del sistema es, por lo general, el maestro que trata al ratón, al teclado, etcétera, como a esclavos. El maestro le dice a los esclavos qué direcciones utilizar, cuándo pueden difundir, durante cuánto tiempo pueden transmitir, qué frecuencias pueden utilizar, etcétera. En el capítulo 4 explicaremos con más detalle el Bluetooth.

El siguiente paso en la conectividad inalámbrica son las LANs inalámbricas. Son sistemas en los que cada computadora tiene un módem de radio y una antena mediante los que se puede comunicar con otros sistemas. En ocasiones, en el techo se coloca una antena con la que las máquinas se comunican, como se ilustra en la figura 1-11(b). Sin embargo, si los sistemas están lo suficientemente cerca, se pueden comunicar de manera directa entre sí en una configuración de

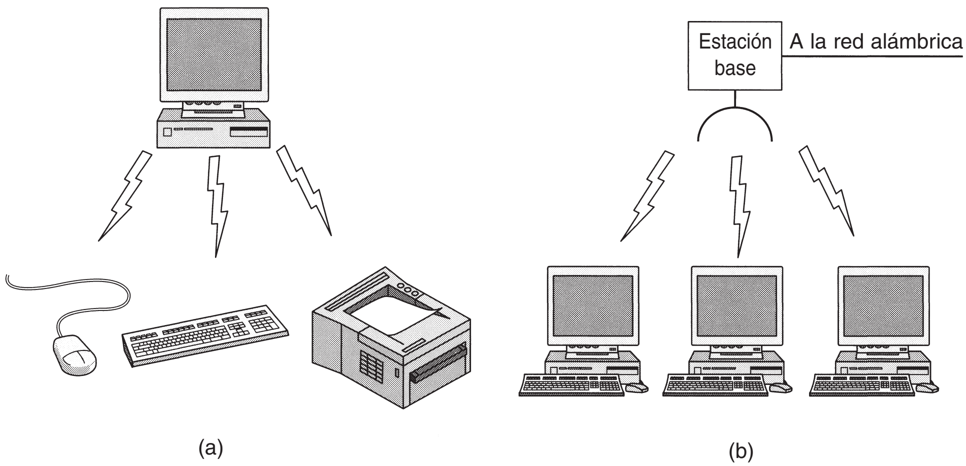


Figura 1-11. (a) Configuración Bluetooth. (b) LAN inalámbrica.

igual a igual. Las LANs inalámbricas se están haciendo cada vez más comunes en casas y oficinas pequeñas, donde instalar Ethernet se considera muy problemático, así como en oficinas ubicadas en edificios antiguos, cafeterías de empresas, salas de conferencias y otros lugares. Existe un estándar para las LANs inalámbricas, llamado **IEEE 802.11,** que la mayoría de los sistemas implementa y que se ha extendido ampliamente. Esto lo explicaremos en el capítulo 4.

El tercer tipo de red inalámbrica se utiliza en sistemas de área amplia. La red de radio utilizada para teléfonos celulares es un ejemplo de un sistema inalámbrico de banda ancha baja. Este sistema ha pasado por tres generaciones. La primera era analógica y sólo para voz. La segunda era digital y sólo para voz. La tercera generación es digital y es tanto para voz como para datos. En cierto sentido, las redes inalámbricas celulares son como las LANs inalámbricas, excepto porque las distancias implicadas son mucho más grandes y las tasas de bits son mucho más bajas. Las LANs inalámbricas pueden funcionar a tasas de hasta 50 Mbps en distancias de decenas de metros. Los sistemas celulares funcionan debajo de 1 Mbps, pero la distancia entre la estación base y la computadora o teléfono se mide en kilómetros más que en metros. En el capítulo 2 hablaremos con mucho detalle sobre estas redes.

Además de estas redes de baja velocidad, también se han desarrollado las redes inalámbricas de área amplia con alto ancho de banda. El enfoque inicial es el acceso inalámbrico a Internet a alta velocidad, desde los hogares y las empresas, dejando a un lado el sistema telefónico. Este servicio se suele llamar servicio de distribución local multipuntos. Lo estudiaremos más adelante. También se ha desarrollado un estándar para éste, llamado IEEE 802.16. Examinaremos dicho estándar en el capítulo 4.

La mayoría de las redes inalámbricas se enlaza a la red alámbrica en algún punto para proporcionar acceso a archivos, bases de datos e Internet. Hay muchas maneras de efectuar estas conexiones, dependiendo de las circunstancias. Por ejemplo, en la figura 1-12(a) mostramos un aeroplano con una serie de personas que utilizan módems y los teléfonos de los respaldos para llamar a la oficina. Cada llamada es independiente de las demás. Sin embargo, una opción mucho

más eficiente es la LAN dentro del avión de la figura 1-12(b), donde cada asiento está equipado con un conector Ethernet al cual los pasajeros pueden acoplar sus computadoras. El avión tiene un solo enrutador, el cual mantiene un enlace de radio con algún enrutador que se encuentre en tierra, y cambia de enrutador conforme avanza el vuelo. Esta configuración es una LAN tradicional, excepto porque su conexión al mundo exterior se da mediante un enlace por radio en lugar de una línea cableada.

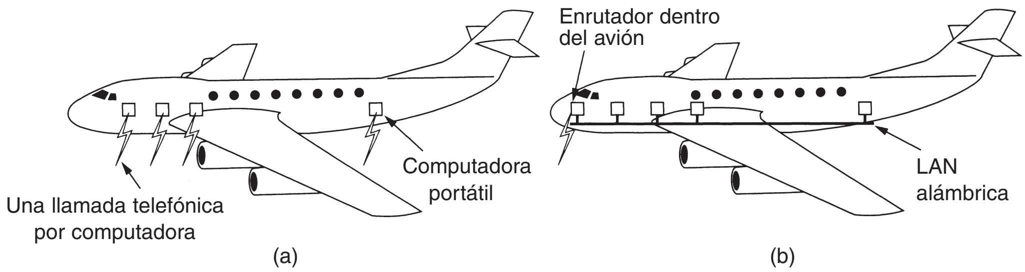


Figura 1-12. (a) Computadoras móviles individuales. (b) LAN dentro del avión.

Muchas personas creen que lo inalámbrico es la onda del futuro (por ejemplo, Bi y cols., 2001; Leeper, 2001; Varshey y Vetter, 2000) pero se ha escuchado una voz disidente. Bob Metcalfe, el inventor de Ethernet, ha escrito: “Las computadoras inalámbricas móviles son como los baños portátiles sin cañería: bacinicas portátiles. Serán muy comunes en los vehículos, en sitios en construcción y conciertos de rock. Mi consejo es que coloque cables en su casa y se quede ahí” (Metcalfe, 1995). La historia podría colocar esta cita en la misma categoría que la explicación de T.J. Watson, presidente de IBM en 1945, de por qué esta empresa no entraba en el negocio de las computadoras: “Cuatro o cinco computadoras deberán ser suficientes para todo el mundo hasta el año 2000”.

1.2.5 Redes domésticas

La conectividad doméstica está en el horizonte. La idea fundamental es que en el futuro la mayoría de los hogares estarán preparados para conectividad de redes. Cualquier dispositivo del hogar será capaz de comunicarse con todos los demás dispositivos y todos podrán accederse por Internet. Éste es uno de esos conceptos visionarios que nadie solicitó (como los controles remotos de TV o los teléfonos celulares), pero una vez que han llegado nadie se puede imaginar cómo habían podido vivir sin ellos.

Muchos dispositivos son capaces de estar conectados en red. Algunas de las categorías más evidentes (con ejemplos) son las siguientes:

1. Computadoras (de escritorio, portátiles, PDAs, periféricos compartidos).
2. Entretenimiento (TV, DVD, VCR, videocámara, cámara fotográfica, estereofónicos, MP3).
3. Telecomunicaciones (teléfono, teléfono móvil, intercomunicadores, fax).
4. Aparatos electrodomésticos (horno de microondas, refrigerador, reloj, horno, aire acondicionado, luces).
5. Telemetría (metro utilitario, alarma contra fuego y robo, termostato, cámaras inalámbricas).

La conectividad de computadoras domésticas ya está aquí, aunque limitada. Muchas casas ya cuentan con un dispositivo para conectar varias computadoras para una conexión rápida a Internet. El entretenimiento por red aún no existe, pero cuanto más y más música y películas se puedan descargar de Internet, habrá más demanda para que los equipos de audio y las televisiones se conecten a Internet. Incluso las personas desearán compartir sus propios vídeos con amigos y familiares, por lo que deberá haber una conexión en ambos sentidos. Los dispositivos de telecomunicaciones ya están conectados al mundo exterior, pero pronto serán digitales y tendrán capacidad de funcionar sobre Internet. Un hogar promedio tal vez tiene una docena de relojes (los de los aparatos electrodomésticos), y todos se tienen que reajustar dos veces al año cuando inicia y termina el tiempo de ahorro de luz de día (horario de verano). Si todos los relojes estuvieran conectados a Internet, ese reajuste se haría en forma automática. Por último, el monitoreo remoto de la casa y su contenido es el probable ganador. Es muy factible que muchos padres deseen invertir en monitorear con sus PDAs a sus bebés dormidos cuando van a cenar fuera de casa, aun cuando contraten a una niñera. Si bien podemos imaginar una red separada para cada área de aplicación, la integración de todas en una sola red es probablemente una mejor idea.

La conectividad doméstica tiene algunas propiedades diferentes a las de otro tipo de redes. Primero, la red y los dispositivos deben ser fáciles de instalar. El autor ha instalado numerosas piezas de hardware y software en varias computadoras durante varios años con resultados diferentes. Al realizar una serie de llamadas telefónicas al personal de soporte técnico del proveedor por lo general recibió respuestas como: 1) Lea el manual; 2) Reinicie la computadora; 3) Elimine todo el hardware y software, excepto los nuestros, y pruebe de nuevo; 4) Descargue de nuestro sitio Web el controlador más reciente y, si todo eso falla, 5) Reformatee el disco duro y reinstale Windows desde el CD-ROM. Decirle al comprador de un refrigerador con capacidad de Internet que descargue e instale una nueva versión del sistema operativo del refrigerador, no conduce a tener clientes contentos. Los usuarios de computadoras están acostumbrados a soportar productos que no funcionan; los clientes que compran automóviles, televisiones y refrigeradores son mucho menos tolerantes. Esperan productos que trabajen al 100% desde que se compran.

Segundo, la red y los dispositivos deben estar plenamente probados en operación. Los equipos de aire acondicionado solían tener una perilla con cuatro parámetros: OFF, LOW, MEDIUM y HIGH (apagado, bajo, medio, alto). Ahora tienen manuales de 30 páginas. Una vez que puedan conectarse en red, no se le haga extraño que tan sólo el capítulo de seguridad tenga 30 páginas. Esto estará más allá de la comprensión de prácticamente todos los usuarios.

Tercero, el precio bajo es esencial para el éxito. Muy pocas personas, si no es que ninguna, pagarán un precio adicional de $50 por un termostato con capacidad de Internet, debido a que no considerarán que monitorear la temperatura de sus casas desde sus trabajos sea algo importante. Tal vez por $5 sí lo comprarían.

Cuarto, la principal aplicación podría implicar multimedia, por lo que la red necesita capacidad suficiente. No hay mercado para televisiones conectadas a Internet que proyecten películas inseguras a una resolución de 320×240 píxeles y 10 cuadros por segundo. Fast Ethernet, el caballo de batalla en la mayoría de las oficinas, no es bastante buena para multimedia. En consecuencia, para que las redes domésticas lleguen a ser productos masivos en el mercado, requerirán mejor desempeño que el de las redes de oficina actuales, así como precios más bajos.

Quinto, se podría empezar con uno o dos dispositivos y expandir de manera gradual el alcance de la red. Esto significa que no habrá problemas con el formato. Decir a los consumidores que adquieran periféricos con interfaces IEEE 1394 (FireWire) y años después retractarse y decir que USB 2.0 es la interfaz del mes, es hacer clientes caprichosos. La interfaz de red tendrá que permanecer estable durante muchos años; el cableado (si lo hay) deberá permanecer estable durante décadas.

Sexto, la seguridad y la confianza serán muy importantes. Perder algunos archivos por un virus de correo electrónico es una cosa; que un ladrón desarme su sistema de seguridad desde su PDA y luego saquee su casa es algo muy diferente.

Una pregunta interesante es si las redes domésticas serán alámbricas o inalámbricas. La mayoría de los hogares ya tiene seis redes instaladas: electricidad, teléfono, televisión por cable, agua, gas y alcantarillado. Agregar una séptima durante la construcción de una casa no es difícil, pero acondicionar las casas existentes para agregar dicha red es costoso. Los costos favorecen la conectividad inalámbrica, pero la seguridad favorece la conectividad alámbrica. El problema con la conectividad inalámbrica es que las ondas de radio que utiliza traspasan las paredes con mucha facilidad. No a todos les gusta la idea de que cuando vaya a imprimir, se tope con la conexión de su vecino y pueda leer el correo electrónico de éste. En el capítulo 8 estudiaremos cómo se puede utilizar la encriptación para proporcionar seguridad, pero en el contexto de una red doméstica la seguridad debe estar bien probada, incluso para usuarios inexpertos. Es más fácil decirlo que hacerlo, incluso en el caso de usuarios expertos.

Para abreviar, la conectividad doméstica ofrece muchas oportunidades y retos. La mayoría de ellos se relaciona con la necesidad de que sean fáciles de manejar, confiables y seguros, en particular en manos de usuarios no técnicos, y que al mismo tiempo proporcionen alto desempeño a bajo costo.

1.2.6 Interredes

Existen muchas redes en el mundo, a veces con hardware y software diferentes. Con frecuencia, las personas conectadas a una red desean comunicarse con personas conectadas a otra red diferente. La satisfacción de este deseo requiere que se conecten diferentes redes, con frecuencia incompatibles, a veces mediante máquinas llamadas **puertas de enlace** (*gateways*) para hacer la conexión y proporcionar la traducción necesaria, tanto en términos de hardware como de software. Un conjunto de redes interconectadas se llama **interred.**

Una forma común de interred es el conjunto de LANs conectadas por una WAN. De hecho, si tuviéramos que reemplazar la etiqueta "subred" en la figura 1-9 por "WAN", no habría nada más que cambiar en la figura. En este caso, la única diferencia técnica real entre una subred y una WAN es si hay *hosts* presentes. Si el sistema que aparece en el área gris contiene solamente enrutadores, es una subred; si contiene enrutadores y *hosts*, es una WAN. Las diferencias reales se relacionan con la propiedad y el uso.

Subredes, redes e interredes con frecuencia se confunden. La subred tiene más sentido en el contexto de una red de área amplia, donde se refiere a un conjunto de enrutadores y líneas de

comunicación poseídas por el operador de redes. Como una analogía, el sistema telefónico consta de oficinas de conmutación telefónica que se conectan entre sí mediante líneas de alta velocidad, y a los hogares y negocios, mediante líneas de baja velocidad. Estas líneas y equipos, poseídas y administradas por la compañía de teléfonos, forman la subred del sistema telefónico. Los teléfonos mismos (los *hosts* en esta analogía) no son parte de la subred. La combinación de una subred y sus *hosts* forma una red. En el caso de una LAN, el cable y los *hosts* forman la red. En realidad, ahí no hay una subred.

Una interred se forma cuando se interconectan redes diferentes. Desde nuestro punto de vista, al conectar una LAN y una WAN o conectar dos LANs se forma una interred, pero existe poco acuerdo en la industria en cuanto a la terminología de esta área. Una regla de oro es que si varias empresas pagaron por la construcción de diversas partes de la red y cada una mantiene su parte, tenemos una interred más que una sola red. Asimismo, si la terminología subyacente es diferente en partes diferentes (por ejemplo, difusión y punto a punto), probablemente tengamos dos redes.

1.3 SOFTWARE DE REDES

Las primeras redes de computadoras se diseñaron teniendo al hardware como punto principal y al software como secundario. Esta estrategia ya no funciona. Actualmente el software de redes está altamente estructurado. En las siguientes secciones examinaremos en detalle la técnica de estructuración de software. El método descrito aquí es la clave de todo el libro y se presentará con mucha frecuencia más adelante.

1.3.1 Jerarquías de protocolos

Para reducir la complejidad de su diseño, la mayoría de las redes está organizada como una pila de **capas** o **niveles,** cada una construida a partir de la que está debajo de ella. El número de capas, así como el nombre, contenido y función de cada una de ellas difieren de red a red. El propósito de cada capa es ofrecer ciertos servicios a las capas superiores, a las cuales no se les muestran los detalles reales de implementación de los servicios ofrecidos.

Este concepto es muy conocido y utilizado en la ciencia computacional, donde se conoce de diversas maneras, como ocultamiento de información, tipos de datos abstractos, encapsulamiento de datos y programación orientada a objetos. La idea básica es que una pieza particular de software (o hardware) proporciona un servicio a sus usuarios pero nunca les muestra los detalles de su estado interno ni sus algoritmos.

La capa n de una máquina mantiene una conversación con la capa n de otra máquina. Las reglas y convenciones utilizadas en esta conversación se conocen de manera colectiva como protocolo de capa n. Básicamente, un **protocolo** es un acuerdo entre las partes en comunicación sobre cómo se debe llevar a cabo la comunicación. Como una analogía, cuando se presenta una mujer con un hombre, ella podría elegir no darle la mano. Él, a su vez, podría decidir saludarla de mano o de beso, dependiendo, por ejemplo, de si es una abogada americana o una princesa europea en

una reunión social formal. Violar el protocolo hará más difícil la comunicación, si no es que imposible.

En la figura 1-13 se ilustra una red de cinco capas. Las entidades que abarcan las capas correspondientes en diferentes máquinas se llaman **iguales** (*peers*). Los iguales podrían ser procesos, dispositivos de hardware o incluso seres humanos. En otras palabras, los iguales son los que se comunican a través del protocolo.

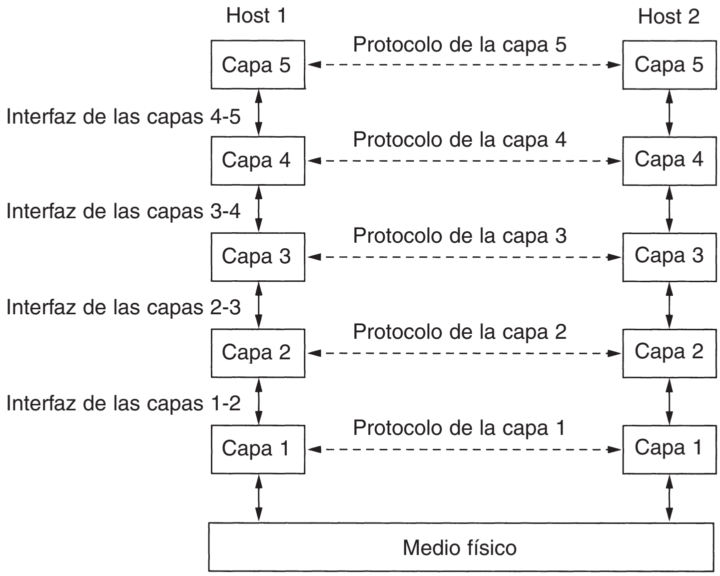


Figura 1-13. Capas, protocolos e interfaces.

En realidad, los datos no se transfieren de manera directa desde la capa *n* de una máquina a la capa *n* de la otra máquina, sino que cada capa pasa los datos y la información de control a la capa inmediatamente inferior, hasta que se alcanza la capa más baja. Debajo de la capa 1 se encuentra el **medio físico** a través del cual ocurre la comunicación real. En la figura 1-13, la comunicación virtual se muestra con líneas punteadas, en tanto que la física, con líneas sólidas.

Entre cada par de capas adyacentes está una **interfaz.** Ésta define qué operaciones y servicios primitivos pone la capa más baja a disposición de la capa superior inmediata. Cuando los diseñadores de redes deciden cuántas capas incluir en una red y qué debe hacer cada una, una de las consideraciones más importantes es definir interfaces limpias entre las capas. Hacerlo así, a su vez, requiere que la capa desempeñe un conjunto específico de funciones bien entendidas. Además de minimizar la cantidad de información que se debe pasar entre las capas, las interfaces bien definidas simplifican el reemplazo de la implementación de una capa con una implementación totalmente diferente (por ejemplo, todas las líneas telefónicas se reemplazan con canales por satélite)

porque todo lo que se pide de la nueva implementación es que ofrezca exactamente el mismo conjunto de servicios a su vecino de arriba, como lo hacía la implementación anterior. De hecho, es muy común que diferentes *hosts* utilicen diferentes implementaciones.

Un conjunto de capas y protocolos se conoce como **arquitectura de red.** La especificación de una arquitectura debe contener información suficiente para permitir que un implementador escriba el programa o construya el hardware para cada capa de modo que se cumpla correctamente con el protocolo apropiado. Ni los detalles de la implementación ni las especificaciones de las interfaces son parte de la arquitectura porque están ocultas en el interior de las máquinas y no son visibles desde el exterior. Incluso, tampoco es necesario que las interfaces de todas las máquinas en una red sean las mismas, siempre y cuando cada máquina pueda utilizar correctamente todos los protocolos. La lista de protocolos utilizados por un sistema, un protocolo por capa, se conoce como **pila de protocolos.** Los aspectos de las arquitecturas de red, las pilas de protocolos y los protocolos mismos son el tema principal de este libro.

Una analogía podría ayudar a explicar la idea de comunicación entre múltiples capas. Imagine a dos filósofos (procesos de iguales en la capa 3), uno de los cuales habla urdu e inglés, y el otro chino y francés. Puesto que no tienen un idioma común, cada uno contrata un traductor (proceso de iguales en la capa 2) y cada uno a su vez contacta a una secretaria (procesos de iguales en la capa 1). El filósofo 1 desea comunicar su afición por el *oryctolagus cuniculus* a su igual. Para eso, le pasa un mensaje (en inglés) a través de la interfaz de las capas 2-3 a su traductor, diciendo: "Me gustan los conejos", como se ilustra en la figura 1-14. Los traductores han acordado un idioma neutral conocido por ambos, el holandés, para que el mensaje se convierta en "Ik vind konijnen leuk". La elección del idioma es el protocolo de la capa 2 y los procesos de iguales de dicha capa son quienes deben realizarla.

Entonces el traductor le da el mensaje a una secretaria para que lo transmita por, digamos, fax (el protocolo de la capa 1). Cuando el mensaje llega, se traduce al francés y se pasa al filósofo 2 a través de la interfaz de las capas 2-3. Observe que cada protocolo es totalmente independiente de los demás en tanto no cambien las interfaces. Los traductores pueden cambiar de holandés a, digamos, finlandés, a voluntad, siempre y cuando los dos estén de acuerdo y no cambien su interfaz con las capas 1 o 3. Del mismo modo, las secretarias pueden cambiar de fax a correo electrónico o teléfono sin molestar (o incluso avisar) a las demás capas. Cada proceso podría agregar alguna información destinada sólo a su igual. Esta información no se pasa a la capa superior.

Ahora veamos un ejemplo más técnico: cómo proporcionar comunicación a la capa superior de la red de cinco capas de la figura 1-15. Un proceso de aplicación que se ejecuta en la capa 5 produce un mensaje, *M,* y lo pasa a la capa 4 para su transmisión.

La capa 4 pone un **encabezado** al frente del mensaje para identificarlo y pasa el resultado a la capa 3. El encabezado incluye información de control, como números de secuencia, para que la capa 4 de la máquina de destino entregue los mensajes en el orden correcto si las capas inferiores no mantienen la secuencia. En algunas capas los encabezados también pueden contener tamaños, medidas y otros campos de control.

En muchas redes no hay límites para el tamaño de mensajes transmitidos en el protocolo de la capa 4, pero casi siempre hay un límite impuesto por el protocolo de la capa 3. En consecuencia,

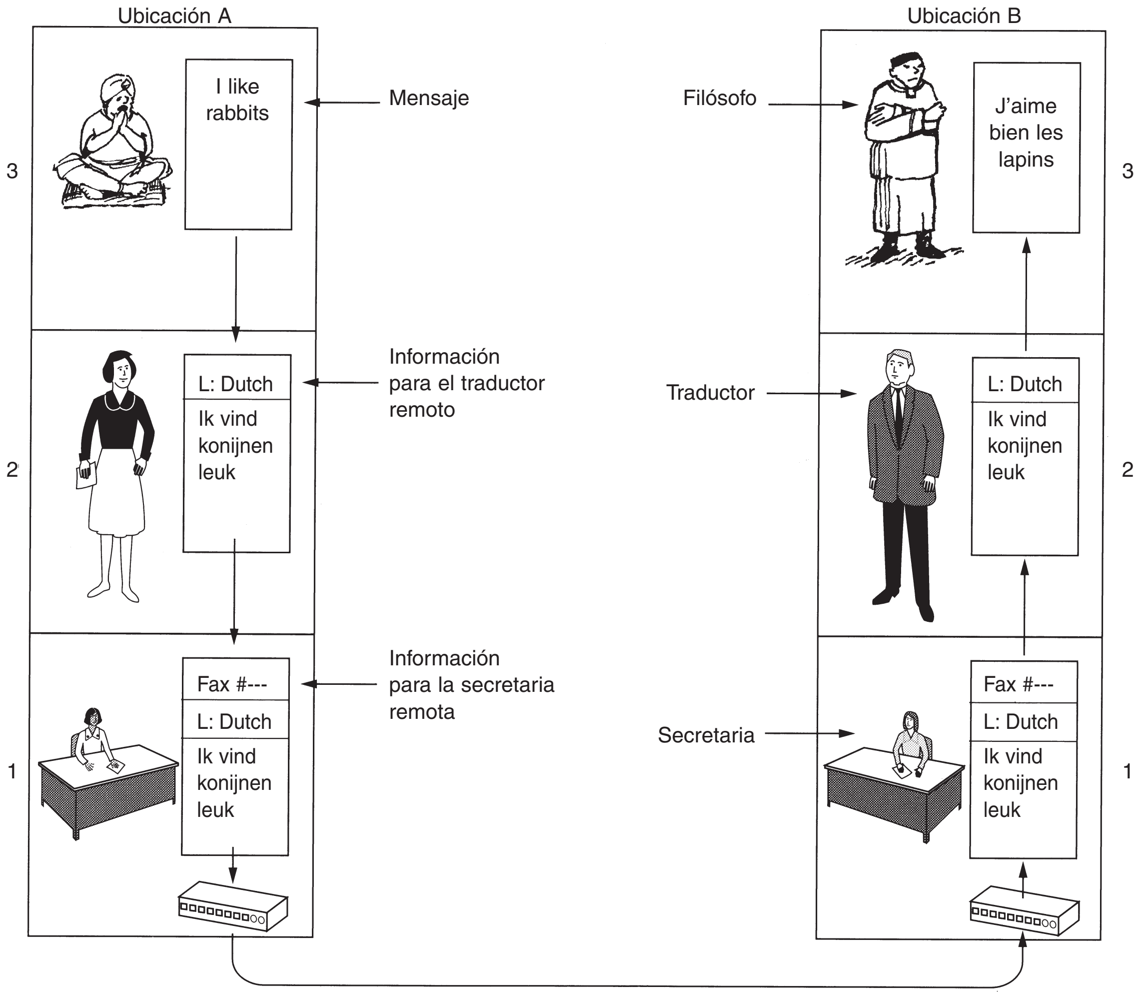


Figura 1-14. Arquitectura filósofo-traductor-secretaria.

la capa 3 debe desintegrar en unidades más pequeñas, paquetes, los mensajes que llegan, y a cada paquete le coloca un encabezado. En este ejemplo, M se divide en dos partes, M_1 y M_2.

La capa 3 decide cuál de las líneas que salen utilizar y pasa los paquetes a la capa 2. Ésta no sólo agrega un encabezado a cada pieza, sino también un terminador, y pasa la unidad resultante a la capa 1 para su transmisión física. En la máquina receptora el mensaje pasa hacia arriba de capa en capa, perdiendo los encabezados conforme avanza. Ninguno de los encabezados de las capas inferiores a n llega a la capa n.

Lo que debe entender en la figura 1-15 es la relación entre las comunicaciones virtual y real, y la diferencia entre protocolos e interfaces. Por ejemplo, los procesos de iguales en la capa 4 piensan

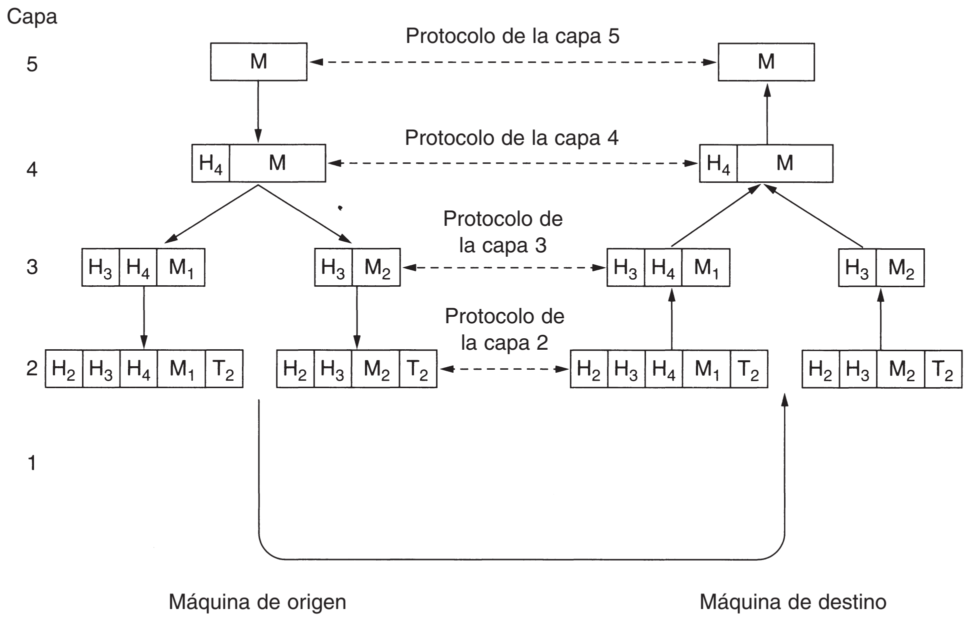


Figura 1-15. Ejemplo de flujo de información que soporta una comunicación virtual en la capa 5.

conceptualmente de su comunicación como si fuera "horizontal", y utilizan el protocolo de la capa 4. Pareciera que cada uno tuviera un procedimiento llamado algo así como *EnviadoalOtroLado* y *RecibidoDesdeElOtroLado*, aun cuando estos procedimientos en realidad se comunican con las capas inferiores a través de la interfaz de las capas 3-4, no con el otro lado.

La abstracción del proceso de iguales es básica para todo diseño de red. Al utilizarla, la inmanejable tarea de diseñar toda la red se puede fragmentar en varios problemas de diseño más pequeños y manejables, es decir, el diseño de las capas individuales.

Aunque la sección 1.3 se llama "Software de redes", vale la pena precisar que las capas inferiores de una jerarquía de protocolos se implementan con frecuencia en el hardware o en el firmware. No obstante, están implicados los algoritmos de protocolo complejos, aun cuando estén integrados (en todo o en parte) en el hardware.

1.3.2 Aspectos de diseño de las capas

Algunos de los aspectos clave de diseño que ocurren en las redes de computadoras están presentes en las diversas capas. Más adelante mencionaremos brevemente algunos de los más importantes.

Cada capa necesita un mecanismo para identificar a los emisores y a los receptores. Puesto que una red por lo general tiene muchas computadoras —algunas de las cuales tienen varios procesos—, se necesita un método para que un proceso en una máquina especifique con cuál de ellas

quiere hablar. Como consecuencia de tener múltiples destinos, se necesita alguna forma de **direccionamiento** a fin de precisar un destino específico.

Otro conjunto de decisiones de diseño concierne a las reglas de la transferencia de datos. En algunos sistemas, los datos viajan sólo en una dirección; en otros, pueden viajar en ambas direcciones. El protocolo también debe determinar a cuántos canales lógicos corresponde la conexión y cuáles son sus prioridades. Muchas redes proporcionan al menos dos canales lógicos por conexión, uno para los datos normales y otro para los urgentes.

El **control de errores** es un aspecto importante porque los circuitos de comunicación física no son perfectos. Muchos códigos de detección y corrección de errores son conocidos, pero los dos extremos de la conexión deben estar de acuerdo en cuál es el que se va a utilizar. Además, el receptor debe tener algún medio de decirle al emisor qué mensajes se han recibido correctamente y cuáles no.

No todos los canales de comunicación conservan el orden en que se les envían los mensajes. Para tratar con una posible pérdida de secuencia, el protocolo debe incluir un mecanismo que permita al receptor volver a unir los pedazos en forma adecuada. Una solución obvia es numerar las piezas, pero esta solución deja abierta la cuestión de qué se debe hacer con las piezas que llegan sin orden.

Un aspecto que ocurre en cada nivel es cómo evitar que un emisor rápido sature de datos a un receptor más lento. Se han propuesto varias soluciones que explicaremos más adelante. Algunas de ellas implican algún tipo de retroalimentación del receptor al emisor, directa o indirectamente, dependiendo de la situación actual del receptor. Otros limitan al emisor a una velocidad de transmisión acordada. Este aspecto se conoce como **control de flujo.**

Otro problema que se debe resolver en algunos niveles es la incapacidad de todos los procesos de aceptar de manera arbitraria mensajes largos. Esta propiedad conduce a mecanismos para desensamblar, transmitir y reensamblar mensajes. Un aspecto relacionado es el problema de qué hacer cuando los procesos insisten en transmitir datos en unidades tan pequeñas que enviarlas por separado es ineficaz. La solución a esto es reunir en un solo mensaje grande varios mensajes pequeños que vayan dirigidos a un destino común y desmembrar dicho mensaje una vez que llegue a su destino.

Cuando es inconveniente o costoso establecer una conexión separada para cada par de procesos de comunicación, la capa subyacente podría decidir utilizar la misma conexión para múltiples conversaciones sin relación entre sí. Siempre y cuando esta **multiplexión** y **desmultiplexión** se realice de manera transparente, cualquier capa la podrá utilizar. La multiplexión se necesita en la capa física, por ejemplo, donde múltiples conversaciones comparten un número limitado de circuitos físicos. Cuando hay múltiples rutas entre el origen y el destino, se debe elegir la mejor o las mejores entre todas ellas. A veces esta decisión se debe dividir en dos o más capas. Por ejemplo, para enviar datos de Londres a Roma, se debe tomar una decisión de alto nivel para pasar por Francia o Alemania, dependiendo de sus respectivas leyes de privacidad. Luego se debe tomar una decisión de bajo nivel para seleccionar uno de los circuitos disponibles dependiendo de la carga de tráfico actual. Este tema se llama **enrutamiento.**

1.3.3 Servicios orientados a la conexión y no orientados a la conexión

Las capas pueden ofrecer dos tipos de servicios a las capas que están sobre ellas: orientados a la conexión y no orientados a la conexión. En esta sección veremos estos dos tipos y examinaremos las diferencias que hay entre ellos.

El **servicio orientado a la conexión** se concibió con base en el sistema telefónico. Para hablar con alguien, usted levanta el teléfono, marca el número, habla y luego cuelga. Del mismo modo, para usar un servicio de red orientado a la conexión, el usuario del servicio primero establece una conexión, la utiliza y luego la abandona. El aspecto esencial de una conexión es que funciona como un tubo: el emisor empuja objetos (bits) en un extremo y el receptor los toma en el otro extremo. En la mayoría de los casos se conserva el orden para que los bits lleguen en el orden en que se enviaron.

En algunos casos, al establecer la conexión, el emisor, el receptor y la subred realizan una **negociación** sobre los parámetros que se van a utilizar, como el tamaño máximo del mensaje, la calidad del servicio solicitado y otros temas. Por lo general, un lado hace una propuesta y el otro la acepta, la rechaza o hace una contrapropuesta.

En contraste, el **servicio no orientado a la conexión** se concibió con base en el sistema postal. Cada mensaje (carta) lleva completa la dirección de destino y cada una se enruta a través del sistema, independientemente de las demás. En general, cuando se envían dos mensajes al mismo destino, el primero que se envíe será el primero en llegar. Sin embargo, es posible que el que se envió primero se dilate tanto que el segundo llegue primero.

Cada servicio se puede clasificar por la **calidad del servicio.** Algunos servicios son confiables en el sentido de que nunca pierden datos. Por lo general, en un servicio confiable el receptor confirma la recepción de cada mensaje para que el emisor esté seguro de que llegó. Este proceso de confirmación de recepción introduce sobrecargas y retardos, que con frecuencia son valiosos pero a veces son indeseables.

Una situación típica en la que un servicio orientado a la conexión es apropiado es en la transferencia de archivos. El propietario del archivo desea estar seguro de que lleguen correctamente todos los bits y en el mismo orden en que se enviaron. Muy pocos clientes que transfieren archivos preferirían un servicio que revuelve o pierde ocasionalmente algunos bits, aunque fuera mucho más rápido.

Un servicio orientado a la conexión confiable tiene dos variantes menores: secuencias de mensaje y flujo de bytes. En la primera variante se conservan los límites del mensaje. Cuando se envían dos mensajes de 1024 bytes, llegan en dos mensajes distintos de 1024 bytes, nunca en un solo mensaje de 2048 bytes. En la segunda, la conexión es simplemente un flujo de bytes, sin límites en el mensaje. Cuando llegan los 2048 bytes al receptor, no hay manera de saber si se enviaron como un mensaje de 2048 bytes o dos mensajes de 1024 bytes o 2048 mensajes de un byte. Si se envían las páginas de un libro en mensajes separados sobre una red a una fotocomponedora, podría ser importante que se conserven los límites de los mensajes. Por otra parte, cuando un usuario inicia sesión en un servidor remoto, todo lo que se necesita es un flujo de bytes desde la computadora del usuario al servidor. Los límites del mensaje no son importantes.

Como lo mencionamos antes, para algunas aplicaciones, los retardos de tránsito ocasionados por las confirmaciones de recepción son inaceptables. Una de estas aplicaciones es el tráfico de voz digitalizada. Es preferible para los usuarios de teléfono escuchar un poco de ruido en la línea de vez en cuando que experimentar un retardo esperando las confirmaciones de recepción. Del mismo modo, tener algunos píxeles erróneos cuando se transmite una videoconferencia no es problema, pero experimentar sacudidas en la imagen cuando se interrumpe el flujo para corregir errores es muy molesto.

No todas las aplicaciones requieren conexiones. Por ejemplo, conforme el correo electrónico se vuelve más común, la basura electrónica también se torna más común. Es probable que el emisor de correo electrónico basura no desee enfrentarse al problema de configurar una conexión y luego desarmarla sólo para enviar un elemento. Tampoco es 100 por ciento confiable enviar lo esencial, sobre todo si eso es más costoso. Todo lo que se necesita es una forma de enviar un mensaje único que tenga una alta, aunque no garantizada, probabilidad de llegar. Al servicio no orientado a la conexión no confiable (es decir, sin confirmación de recepción) se le conoce como **servicio de datagramas,** en analogía con el servicio de telegramas, que tampoco devuelve una confirmación de recepción al emisor.

En otras situaciones se desea la conveniencia de no tener que establecer una conexión para enviar un mensaje corto, pero la confiabilidad es esencial. Para estas aplicaciones se puede proporcionar el **servicio de datagramas confirmados.** Es como enviar una carta certificada y solicitar una confirmación de recepción. Cuando ésta regresa, el emisor está absolutamente seguro de que la carta se ha entregado a la parte destinada y no se ha perdido durante el trayecto.

Otro servicio más es el de **solicitud-respuesta.** En este servicio el emisor transmite un solo datagrama que contiene una solicitud; a continuación el servidor envía la respuesta. Por ejemplo, una solicitud a la biblioteca local preguntando dónde se habla uighur cae dentro de esta categoría. El esquema de solicitud-respuesta se usa comúnmente para implementar la comunicación en el modelo cliente-servidor: el cliente emite una solicitud y el servidor la responde. La figura 1-16 resume los tipos de servicios que se acaban de exponer.

	Servicio	Ejemplo
Orientado a la conexión	Flujo confiable de mensajes	Secuencia de páginas
	Flujo confiable de bytes	Inicio de sesión remoto
	Conexión no confiable	Voz digitalizada
No orientado a la conexión	Datagrama no confiable	Correo electrónico basura
	Datagrama confirmado	Correo certificado
	Solicitud-respuesta	Consulta de base de datos

Figura 1-16. Seis tipos de servicio diferentes.

El concepto del uso de la comunicación no confiable podría ser confuso al principio. Después de todo, en realidad, ¿por qué preferiría alguien la comunicación no confiable a la comunicación

confiable? Antes que nada, la comunicación confiable (en nuestro sentido, es decir, con confirmación de la recepción) podría no estar disponible. Por ejemplo, Ethernet no proporciona comunicación confiable. Ocasionalmente, los paquetes se pueden dañar en el tránsito. Toca al protocolo más alto enfrentar este problema. En segundo lugar, los retardos inherentes al servicio confiable podrían ser inaceptables, en particular para aplicaciones en tiempo real como multimedia. Éstas son las razones de que coexistan la comunicación no confiable y la confiable.

1.3.4 Primitivas de servicio

Un servicio se especifica formalmente como un conjunto de **primitivas** (operaciones) disponibles a un proceso de usuario para que acceda al servicio. Estas primitivas le indican al servicio que desempeñe alguna acción o reporte sobre una acción que ha tomado una entidad igual. Si la pila de protocolos se ubica en el sistema operativo, como suele suceder, por lo general las primitivas son llamadas al sistema. Estas llamadas provocan un salto al modo de *kernel*, que entonces cede el control de la máquina al sistema operativo para enviar los paquetes necesarios.

El conjunto de primitivas disponible depende de la naturaleza del servicio que se va a proporcionar. Las primitivas de servicio orientado a la conexión son diferentes de las del servicio no orientado a la conexión. Como un ejemplo mínimo de las primitivas para servicio que se podrían proporcionar para implementar un flujo de bytes confiable en un ambiente cliente-servidor, considere las primitivas listadas en la figura 1-17.

Primitiva	Significado
LISTEN	Bloquea en espera de una conexión entrante
CONNECT	Establece una conexión con el igual en espera
RECEIVE	Bloquea en espera de un mensaje entrante
SEND	Envía un mensaje al igual
DISCONNECT	Da por terminada una conexión

Figura 1-17. Cinco primitivas de servicio para la implementación de un servicio simple orientado a la conexión.

Estas primitivas se podrían usar como sigue. En primer lugar, el servidor ejecuta LISTEN para indicar que está preparado para aceptar las conexiones entrantes. Una manera común de implementar LISTEN es hacer que bloquee la llamada al sistema. Después de ejecutar la primitiva, el proceso del servidor se bloquea hasta que aparece una solicitud de conexión.

A continuación, el proceso del cliente ejecuta CONNECT para establecer una conexión con el servidor. La llamada CONNECT necesita especificar a quién conecta con quién, así que podría tener un parámetro que diera la dirección del servidor. El sistema operativo, en general, envía un paquete al igual solicitándole que se conecte, como se muestra en (1) en la figura 1-18. El proceso del cliente se suspende hasta que haya una respuesta. Cuando el paquete llega al servidor, es procesado ahí por el sistema operativo. Cuando el sistema ve que el paquete es una solicitud de

conexión, verifica si hay un escuchador. En ese caso hace dos cosas: desbloquea al escuchador y envía de vuelta una confirmación de recepción (2). La llegada de esta confirmación libera entonces al cliente. En este punto tanto el cliente como el servidor están en ejecución y tienen establecida una conexión. Es importante observar que la confirmación de recepción (2) es generada por el código del protocolo mismo, no en respuesta a una primitiva al nivel de usuario. Si llega una solicitud de conexión y no hay un escuchador, el resultado es indefinido. En algunos sistemas el paquete podría ser puesto en cola durante un breve tiempo en espera de un LISTEN.

La analogía obvia entre este protocolo y la vida real es un consumidor (cliente) que llama al gerente de servicios a clientes de una empresa. El gerente de servicios empieza por estar cerca del teléfono en caso de que éste suene. Entonces el cliente hace la llamada. Cuando el gerente levanta el teléfono se establece la conexión.

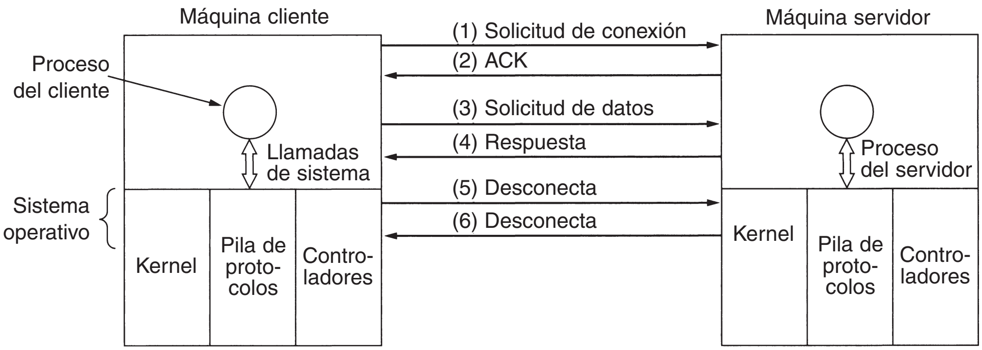


Figura 1-18. Paquetes enviados en una interacción simple cliente-servidor sobre una red orientada a la conexión.

El paso siguiente es que el servidor ejecute RECEIVE para prepararse para aceptar la primera solicitud. Normalmente, el servidor hace esto de inmediato en cuanto está libre de LISTEN, antes de que la confirmación de recepción pueda volver al cliente. La llamada RECEIVE bloquea al servidor.

Entonces el cliente ejecuta SEND para transmitir sus solicitudes (3) seguidas de la ejecución de RECEIVE para obtener la respuesta.

La llegada del paquete de solicitud a la máquina servidor desbloquea el proceso del servidor para que pueda procesar la solicitud. Una vez hecho su trabajo, utiliza SEND para devolver la respuesta al cliente (4). La llegada de este paquete desbloquea al cliente, que ahora puede revisar la respuesta. Si el cliente tiene solicitudes adicionales las puede hacer ahora. Si ha terminado, puede utilizar DISCONNECT para finalizar la conexión. Por lo común, un DISCONNECT inicial es una llamada de bloqueo, que suspende al cliente y envía un paquete al servidor en el cual le indica que ya no es necesaria la conexión (5). Cuando el servidor recibe el paquete también emite un DISCONNECT, enviando la confirmación de recepción al cliente y terminando la conexión. Cuando el paquete del servidor (6) llega a la máquina cliente, el proceso del cliente se libera y finaliza la conexión. En pocas palabras, ésta es la manera en que funciona la comunicación orientada a la conexión.

Desde luego, no todo es tan sencillo. Hay muchas cosas que pueden fallar. La temporización puede estar mal (por ejemplo, CONNECT se hace antes de LISTEN), se pueden perder paquetes, etcétera. Más adelante veremos en detalle estos temas, pero por el momento la figura 1-18 resume cómo podría funcionar la comunicación cliente-servidor en una red orientada a la conexión.

Dado que se requieren seis paquetes para completar este protocolo, cabría preguntarse por qué no se usa en su lugar un protocolo no orientado a la conexión. La respuesta es que en un mundo perfecto podría utilizarse, en cuyo caso bastarían dos paquetes: uno para la solicitud y otro para la respuesta. Sin embargo, en el caso de mensajes grandes en cualquier dirección (por ejemplo, en un archivo de megabytes), errores de transmisión y paquetes perdidos, la situación cambia. Si la respuesta constara de cientos de paquetes, algunos de los cuales se podrían perder durante la transmisión, ¿cómo sabría el cliente si se han perdido algunas piezas? ¿Cómo podría saber que el último paquete que recibió fue realmente el último que se envió? Suponga que el cliente esperaba un segundo archivo. ¿Cómo podría saber que el paquete 1 del segundo archivo de un paquete 1 perdido del primer archivo que de pronto apareció va en camino al cliente? Para abreviar, en el mundo real un simple protocolo de solicitud-respuesta en una red no confiable suele ser inadecuado. En el capítulo 3 estudiaremos en detalle una variedad de protocolos que soluciona éstos y otros problemas. Por el momento, baste decir que a veces es muy conveniente tener un flujo de bytes ordenado y confiable entre procesos.

1.3.5 Relación de servicios a protocolos

Servicios y protocolos son conceptos distintos, aunque con frecuencia se confunden. Sin embargo, esta distinción es tan importante que por esa razón ponemos énfasis de nuevo en ese punto. Un *servicio* es un conjunto de primitivas (operaciones) que una capa proporciona a la capa que está sobre ella. El servicio define qué operaciones puede realizar la capa en beneficio de sus usuarios, pero no dice nada de cómo se implementan tales operaciones. Un servicio está relacionado con la interfaz entre dos capas, donde la capa inferior es la que provee el servicio y la superior, quien lo recibe.

Un *protocolo,* en contraste, es un conjunto de reglas que rigen el formato y el significado de los paquetes, o mensajes, que se intercambiaron las entidades iguales en una capa. Las entidades utilizan protocolos para implementar sus definiciones del servicio. Son libres de cambiar sus protocolos cuando lo deseen, siempre y cuando no cambie el servicio visible a sus usuarios. De esta manera, el servicio y el protocolo no dependen uno del otro.

En otras palabras, los servicios se relacionan con las interacciones entre capas, como se ilustra en la figura 1-19. En contraste, los protocolos se relacionan con los paquetes enviados entre entidades iguales de máquinas diferentes. Es importante no confundir estos dos conceptos.

Vale la pena hacer una analogía con los lenguajes de programación. Un servicio es como un tipo de datos abstractos o un objeto en un lenguaje orientado a objetos. Define operaciones que se deben realizar en un objeto pero no especifica cómo se implementan estas operaciones. Un protocolo se relaciona con la *implementación* del servicio y, como tal, el usuario del servicio no puede verlo.

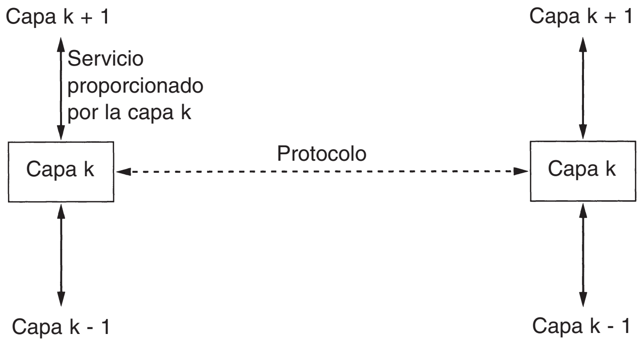


Figura 1-19. La relación entre un servicio y un protocolo.

Muchos protocolos antiguos no distinguían el servicio del protocolo. En efecto, una capa típica podría haber tenido una primitiva de servicio SEND PACKET y el usuario proveía un apuntador a un paquete ensamblado totalmente. Este arreglo significa que el usuario podía ver de inmediato todos los cambios del protocolo. En la actualidad, la mayoría de los diseñadores de redes señalan a este tipo de diseño como un error grave.

1.4 MODELOS DE REFERENCIA

Ahora que hemos visto en teoría las redes con capas, es hora de ver algunos ejemplos. En las dos secciones siguientes veremos dos arquitecturas de redes importantes: los modelos de referencia OSI y TCP/IP. Aunque los *protocolos* asociados con el modelo OSI ya casi no se usan, el *modelo* en sí es muy general y aún es válido, y las características tratadas en cada capa aún son muy importantes. El modelo TCP/IP tiene las propiedades opuestas: el modelo en sí no se utiliza mucho pero los protocolos sí. Por estas razones analizaremos con detalle ambos modelos. Además, a veces podemos aprender más de las fallas que de los aciertos.

1.4.1 El modelo de referencia OSI

El modelo OSI se muestra en la figura 1-20 (sin el medio físico). Este modelo está basado en una propuesta desarrollada por la ISO (Organización Internacional de Estándares) como un primer paso hacia la estandarización internacional de los protocolos utilizados en varias capas (Day y Zimmermann, 1983). Fue revisado en 1995 (Day, 1995). El modelo se llama **OSI** (**Interconexión de Sistemas Abiertos**) de **ISO** porque tiene que ver con la conexión de sistemas abiertos, es decir, sistemas que están abiertos a la comunicación con otros sistemas. Para abreviar, lo llamaremos modelo OSI.

El modelo OSI tiene siete capas. Podemos resumir brevemente los principios que se aplicaron para llegar a dichas capas:

1. Una capa se debe crear donde se necesite una abstracción diferente.
2. Cada capa debe realizar una función bien definida.
3. La función de cada capa se debe elegir con la intención de definir protocolos estandarizados internacionalmente.
4. Los límites de las capas se deben elegir a fin de minimizar el flujo de información a través de las interfaces.
5. La cantidad de capas debe ser suficientemente grande para no tener que agrupar funciones distintas en la misma capa y lo bastante pequeña para que la arquitectura no se vuelva inmanejable.

A continuación analizaremos una por una cada capa del modelo, comenzando con la capa inferior. Observe que el modelo OSI no es en sí una arquitectura de red, debido a que no especifica los servicios y protocolos exactos que se utilizarán en cada capa. Sólo indica lo que debe hacer cada capa. Sin embargo, ISO también ha producido estándares para todas las capas, aunque éstos no son parte del modelo de referencia mismo. Cada uno se ha publicado como un estándar internacional separado.

La capa física

En esta capa se lleva a cabo la transmisión de bits puros a través de un canal de comunicación. Los aspectos del diseño implican asegurarse de que cuando un lado envía un bit 1, éste se reciba en el otro lado como tal, no como bit 0. Las preguntas típicas aquí son: ¿cuántos voltios se deben emplear para representar un 1 y cuántos para representar un 0?, ¿cuántos nanosegundos dura un bit?, ¿la transmisión se debe llevar a cabo en ambas direcciones al mismo tiempo?, ¿cómo se establece la conexión inicial y cómo se finaliza cuando ambos lados terminan?, ¿cuántos pines tiene un conector de red y para qué se utiliza cada uno? Los aspectos de diseño tienen que ver mucho con interfaces mecánicas, eléctricas y de temporización, además del medio físico de transmisión, que está bajo la capa física.

La capa de enlace de datos

La tarea principal de esta capa es transformar un medio de transmisión puro en una línea de comunicación que, al llegar a la capa de red, aparezca libre de errores de transmisión. Logra esta tarea haciendo que el emisor fragmente los datos de entrada en **tramas de datos** (típicamente, de algunos cientos o miles de bytes) y transmitiendo las tramas de manera secuencial. Si el servicio es confiable, el receptor confirma la recepción correcta de cada trama devolviendo una **trama de confirmación de recepción.**

Otra cuestión que surge en la capa de enlace de datos (y en la mayoría de las capas superiores) es cómo hacer que un transmisor rápido no sature de datos a un receptor lento. Por lo general se necesita un mecanismo de regulación de tráfico que indique al transmisor cuánto espacio de búfer tiene el receptor en ese momento. Con frecuencia, esta regulación de flujo y el manejo de errores están integrados.

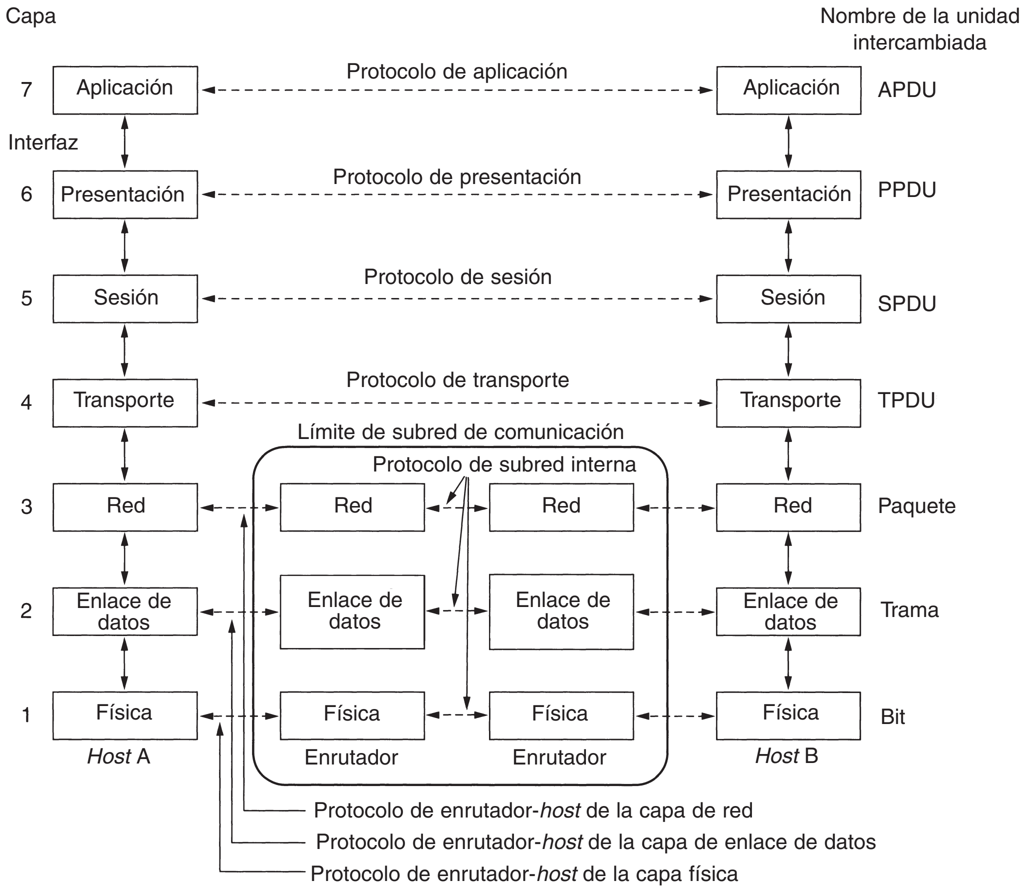


Figura 1-20. El modelo de referencia OSI.

Las redes de difusión tienen un aspecto adicional en la capa de enlace de datos: cómo controlar el acceso al canal compartido. Una subcapa especial de la capa de enlace de datos, la subcapa de control de acceso al medio, se encarga de este problema.*

La capa de red

Esta capa controla las operaciones de la subred. Un aspecto clave del diseño es determinar cómo se enrutan los paquetes desde su origen a su destino. Las rutas pueden estar basadas en tablas estáticas (enrutamiento estático) codificadas en la red y que rara vez cambian.**

*En esta capa se define el direccionamiento físico, que permite a los *hosts* identificar las tramas destinadas a ellos. Este direccionamiento es único, identifica el hardware de red que se está usando y el fabricante, y no se puede cambiar. (N. del R.T.)

**En el enrutamiento estático la ruta que seguirán los paquetes hacia un destino particular es determinada por el administrador de la red. Las rutas también pueden determinarse cuando los enrutadores intercambian información de enrutamiento (enrutamiento dinámico). En este tipo de enrutamiento los enrutadores deciden la ruta que seguirán los paquetes hacia un destino sin la intervención del administrador de red. En el enrutamiento dinámico las rutas pueden cambiar para reflejar la topología o el estado de la red. (N. del R.T.)

Si hay demasiados paquetes en la subred al mismo tiempo, se interpondrán en el camino unos y otros, lo que provocará que se formen cuellos de botella. La responsabilidad de controlar esta congestión también pertenece a la capa de red, aunque esta responsabilidad también puede ser compartida por la capa de transmisión. De manera más general, la calidad del servicio proporcionado (retardo, tiempo de tránsito, inestabilidad, etcétera) también corresponde a la capa de red.

Cuando un paquete tiene que viajar de una red a otra para llegar a su destino, pueden surgir muchos problemas. El direccionamiento utilizado por la segunda red podría ser diferente del de la primera.* La segunda podría no aceptar todo el paquete porque es demasiado largo. Los protocolos podrían ser diferentes, etcétera. La capa de red tiene que resolver todos estos problemas para que las redes heterogéneas se interconecten.

En las redes de difusión, el problema de enrutamiento es simple, por lo que la capa de red a veces es delgada o, en ocasiones, ni siquiera existe.

La capa de transporte

La función básica de esta capa es aceptar los datos provenientes de las capas superiores, dividirlos en unidades más pequeñas si es necesario, pasar éstas a la capa de red y asegurarse de que todas las piezas lleguen correctamente al otro extremo. Además, todo esto se debe hacer con eficiencia y de manera que aísle a las capas superiores de los cambios inevitables en la tecnología del hardware.

La capa de transporte también determina qué tipo de servicio proporcionar a la capa de sesión y, finalmente, a los usuarios de la red. El tipo de conexión de transporte más popular es un canal punto a punto libre de errores que entrega mensajes o bytes en el orden en que se enviaron. Sin embargo, otros tipos de servicio de transporte posibles son la transportación de mensajes aislados, que no garantiza el orden de entrega, y la difusión de mensajes a múltiples destinos. El tipo de servicio se determina cuando se establece la conexión. (Como observación, es imposible alcanzar un canal libre de errores; lo que se quiere dar a entender con este término es que la tasa de error es tan baja que se puede ignorar en la práctica.)

La capa de transporte es una verdadera conexión de extremo a extremo, en toda la ruta desde el origen hasta el destino. En otras palabras, un programa en la máquina de origen lleva a cabo una conversación con un programa similar en la máquina de destino, usando los encabezados de mensaje y los mensajes de control. En las capas inferiores, los protocolos operan entre cada máquina y sus vecinos inmediatos, y no entre las máquinas de los extremos, la de origen y la de destino, las cuales podrían estar separadas por muchos enrutadores. En la figura 1-20 se muestra la diferencia entre las capas 1 a 3, que están encadenadas, y las capas 4 a 7, que operan de extremo a extremo.

La capa de sesión

Esta capa permite que los usuarios de máquinas diferentes establezcan **sesiones** entre ellos. Las sesiones ofrecen varios servicios, como el **control de diálogo** (dar seguimiento de a quién le toca

*El direccionamiento usado en esta capa es un direccionamiento lógico, diferente al direccionamiento físico empleado en la capa de enlace de datos. Este direccionamiento lógico permite que una interfaz o puerto pueda tener más de una dirección de capa de red. (N. del R.T.)

transmitir), **administración de token** (que impide que las dos partes traten de realizar la misma operación crítica al mismo tiempo) y **sincronización** (la adición de puntos de referencia a transmisiones largas para permitirles continuar desde donde se encontraban después de una caída).

La capa de presentación

A diferencia de las capas inferiores, a las que les corresponde principalmente mover bits, a la **capa de presentación** le corresponde la sintaxis y la semántica de la información transmitida. A fin de que las computadoras con diferentes representaciones de datos se puedan comunicar, las estructuras de datos que se intercambiarán se pueden definir de una manera abstracta, junto con una codificación estándar para su uso "en el cable". La capa de presentación maneja estas estructuras de datos abstractas y permite definir e intercambiar estructuras de datos de un nivel más alto (por ejemplo, registros bancarios).

La capa de aplicación

Esta capa contiene varios protocolos que los usuarios requieren con frecuencia. Un protocolo de aplicación de amplio uso es **HTTP** (**Protocolo de Transferencia de Hipertexto),** que es la base de World Wide Web. Cuando un navegador desea una página Web, utiliza este protocolo para enviar al servidor el nombre de dicha página. A continuación, el servidor devuelve la página. Otros protocolos de aplicación se utilizan para la transferencia de archivos, correo electrónico y noticias en la red.

1.4.2 El modelo de referencia TCP/IP

Tratemos ahora el modelo de referencia usado en la abuela de todas las redes de computadoras de área amplia, ARPANET, y en su sucesora, la Internet mundial. Aunque daremos más adelante una breve historia de ARPANET, es útil mencionar algunos de sus aspectos ahora. ARPANET fue una red de investigación respaldada por el DoD (Departamento de Defensa de Estados Unidos). Con el tiempo, conectó cientos de universidades e instalaciones gubernamentales mediante líneas telefónicas alquiladas. Posteriormente, cuando se agregaron redes satelitales y de radio, los protocolos existentes tuvieron problemas para interactuar con ellas, por lo que se necesitaba una nueva arquitectura de referencia. De este modo, la capacidad para conectar múltiples redes en una manera sólida fue una de las principales metas de diseño desde sus inicios. Más tarde, esta arquitectura se llegó a conocer como el **modelo de referencia TCP/IP,** de acuerdo con sus dos protocolos primarios. Su primera definición fue en (Cerf y Kahn, 1974). Posteriormente se definió en (Leiner y cols., 1985). La filosofía del diseño que respalda al modelo se explica en (Clark, 1988).

Ante el temor del DoD de que algunos de sus valiosos *hosts*, enrutadores y puertas de enlace de interredes explotaran en un instante, otro objetivo fue que la red pudiera sobrevivir a la pérdida de hardware de la subred, sin que las conversaciones existentes se interrumpieran. En otras palabras, el DoD quería que las conexiones se mantuvieran intactas en tanto las máquinas de origen

y destino estuvieran funcionando, aunque algunas de las máquinas o líneas de transmisión intermedias quedaran fuera de operación repentinamente. Además, se necesitaba una arquitectura flexible debido a que se preveían aplicaciones con requerimientos divergentes, desde transferencia de archivos a transmisión de palabras en tiempo real.

La capa de interred

Todos estos requerimientos condujeron a la elección de una red de conmutación de paquetes basada en una capa de interred no orientada a la conexión. Esta capa, llamada **capa de interred,** es la pieza clave que mantiene unida a la arquitectura. Su trabajo es permitir que los *hosts* inyecten paquetes dentro de cualquier red y que éstos viajen a su destino de manera independiente (podría ser en una red diferente). Tal vez lleguen en un orden diferente al que fueron enviados, en cuyo caso las capas más altas deberán ordenarlos, si se desea una entrega ordenada. Observe que aquí el concepto "interred" se utiliza en un sentido genérico, aun cuando esta capa se presente en Internet.

Aquí la analogía es con el sistema de correo tradicional. Una persona puede depositar una secuencia de cartas internacionales en un buzón y, con un poco de suerte, la mayoría de ellas se entregará en la dirección correcta del país de destino. Es probable que durante el trayecto, las cartas viajen a través de una o más puertas de enlace de correo internacional, pero esto es transparente para los usuarios. Además, para los usuarios también es transparente el hecho de que cada país (es decir, cada red) tiene sus propios timbres postales, tamaños preferidos de sobre y reglas de entrega.

La capa de interred define un paquete de formato y protocolo oficial llamado **IP** (**Protocolo de Internet**)**.** El trabajo de la capa de interred es entregar paquetes IP al destinatario. Aquí, el enrutamiento de paquetes es claramente el aspecto principal, con el propósito de evitar la congestión. Por estas razones es razonable decir que la capa de interred del modelo TCP/IP es similar en funcionalidad a la capa de red del modelo OSI. La figura 1-21 muestra esta correspondencia.

La capa de transporte

La capa que está arriba de la capa de interred en el modelo TCP/IP se llama **capa de transporte.** Está diseñada para permitir que las entidades iguales en los *hosts* de origen y destino puedan llevar a cabo una conversación, tal como lo hace la capa de transporte OSI. Aquí se han definido dos protocolos de transporte de extremo a extremo. El primero, **TCP** (**Protocolo de Control de Transmisión**)**,** es un protocolo confiable, orientado a la conexión, que permite que un flujo de bytes que se origina en una máquina se entregue sin errores en cualquier otra máquina en la interred. Divide el flujo de bytes entrantes en mensajes discretos y pasa cada uno de ellos a la capa de interred. En el destino, el proceso TCP receptor reensambla en el flujo de salida los mensajes recibidos. TCP también maneja el control de flujo para asegurarse de que un emisor rápido no sature a un receptor lento con más mensajes de los que puede manejar.

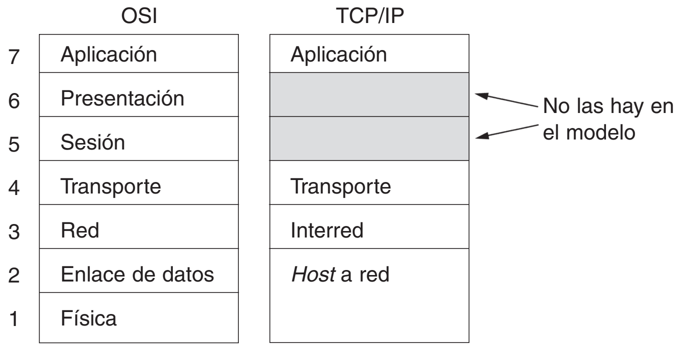


Figura 1-21. El modelo de referencia TCP/IP.

El segundo protocolo de esta capa, **UDP** (**Protocolo de Datagrama de Usuario),** es un protocolo no confiable y no orientado a la conexión para aplicaciones que no desean la secuenciación o el control de flujo de TCP y que desean proporcionar el suyo. También tiene un amplio uso en consultas únicas de solicitud-respuesta de tipo cliente-servidor en un solo envío, así como aplicaciones en las que la entrega puntual es más importante que la precisa, como en la transmisión de voz o vídeo. La relación de IP, TCP y UDP se muestra en la figura 1-22. Puesto que el modelo se desarrolló, se ha implementado IP en muchas otras redes.

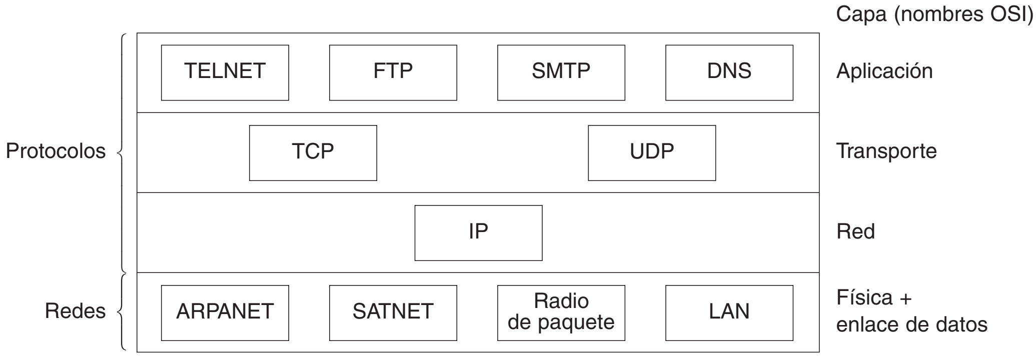


Figura 1-22. Protocolos y redes en el modelo TCP/IP inicialmente.

La capa de aplicación

El modelo TCP/IP no tiene capas de sesión ni de presentación. No se han necesitado, por lo que no se incluyen. La experiencia con el modelo OSI ha probado que este punto de vista es correcto: son de poco uso para la mayoría de las aplicaciones.

Arriba de la capa de transporte está la **capa de aplicación.** Contiene todos los protocolos de nivel más alto. Los primeros incluyeron una terminal virtual (TELNET), transferencia de archivos (FTP) y correo electrónico (SMTP), como se muestra en la figura 1-22. El protocolo de terminal virtual permite que un usuario en una máquina se registre en una máquina remota y trabaje ahí. El protocolo de transferencia de archivos proporciona una manera de mover con eficiencia datos de una máquina a otra. El correo electrónico era originalmente sólo un tipo de transferencia de archivos, pero más tarde se desarrolló un protocolo especializado (SMTP) para él. Con el tiempo, se han agregado muchos otros protocolos: DNS (Sistema de Nombres de Dominio) para la resolución de nombres de *host* en sus direcciones de red; NNTP, para transportar los artículos de noticias de USENET; HTTP, para las páginas de World Wide Web, y muchos otros.

La capa *host* a red

Debajo de la capa de interred hay un gran vacío. El modelo de referencia TCP/IP en realidad no dice mucho acerca de lo que pasa aquí, excepto que puntualiza que el *host* se tiene que conectar a la red mediante el mismo protocolo para que le puedan enviar paquetes IP. Este protocolo no está definido y varía de un *host* a otro y de una red a otra. Este tema rara vez se trata en libros y artículos sobre TCP/IP.

1.4.3 Comparación entre los modelos de referencia OSI y TCP/IP

Los modelos de referencia OSI y TCP/IP tienen mucho en común. Los dos se basan en el concepto de una pila de protocolos independientes. Asimismo, la funcionalidad de las capas es muy parecida. Por ejemplo, en ambos modelos las capas que están arriba de, incluyendo a, la capa de transporte están ahí para proporcionar un servicio de transporte independiente de extremo a extremo a los procesos que desean comunicarse. Estas capas forman el proveedor de transporte. De nuevo, en ambos modelos, las capas que están arriba de la de transporte son usuarias orientadas a la aplicación del servicio de transporte.

A pesar de estas similitudes fundamentales, los dos modelos también tienen muchas diferencias. En esta sección nos enfocaremos en las diferencias clave entre estos dos modelos de referencia. Es importante tener en cuenta que estamos comparando los *modelos de referencia,* no las *pilas de protocolos* correspondientes. Más adelante explicaremos los protocolos. Si desea un libro dedicado a comparar y contrastar TCP/IP y OSI, vea (Piscitello y Chapin, 1993).

Tres conceptos son básicos para el modelo OSI:

1. Servicios.
2. Interfaces.
3. Protocolos.

Probablemente la contribución más grande del modelo OSI es que hace explícita la distinción entre estos tres conceptos. Cada capa desempeña algunos servicios para la capa que está arriba de ella. La definición de *servicio* indica qué hace la capa, no la forma en que la entidad superior tiene acceso a ella, o cómo funciona dicha capa. Define el aspecto semántico de la capa.

La *interfaz* de una capa indica a los procesos que están sobre ella cómo accederla. Especifica cuáles son los parámetros y qué resultados se esperan. Incluso, no dice nada sobre cómo funciona internamente la capa.

Por último, una capa es quien debe decidir qué *protocolos* de iguales utilizar. Puede usar cualesquier protocolos que desee, en tanto consiga que se haga el trabajo (es decir, proporcione los servicios ofrecidos). También puede cambiarlos cuando desee sin afectar el software de las capas superiores.

Estas ideas encajan muy bien con las ideas modernas sobre la programación orientada a objetos. Un objeto, como una capa, cuenta con un conjunto de métodos (operaciones) que pueden ser invocados por procesos que no estén en dicho objeto. La semántica de estos métodos define el conjunto de servicios que ofrece el objeto. Los parámetros y resultados de los métodos forman la interfaz del objeto. El código interno del objeto es su protocolo y no es visible o no tiene importancia fuera del objeto.

Originalmente, el modelo TCP/IP no distinguía entre servicio, interfaz y protocolo, aunque las personas han tratado de readaptarlo con el propósito de hacerlo más parecido al OSI. Por ejemplo, los únicos servicios ofrecidos realmente por la capa de interred son SEND IP PACKET y RECEIVE IP PACKET.

Como consecuencia, los protocolos del modelo OSI están mejor ocultos que los del modelo TCPI/IP y se pueden reemplazar fácilmente conforme cambia la tecnología. La facilidad para realizar tales cambios es uno de los objetivos principales de tener protocolos en capas.

El modelo de referencia OSI se vislumbró *antes* de que se inventaran los protocolos correspondientes. Esta clasificación significa que el modelo no estaba diseñado para un conjunto particular de protocolos, un hecho que lo hizo general. Una deficiencia de esta clasificación es que los diseñadores no tenían mucha experiencia con el asunto y no tenían una idea concreta de qué funcionalidad poner en qué capa.

Por ejemplo, originalmente la capa de enlace de datos sólo trataba con redes de punto a punto. Cuando llegaron las redes de difusión, se tuvo que extender una nueva subcapa en el modelo. Cuando las personas empezaron a construir redes reales utilizando el modelo OSI y los protocolos existentes, se descubrió que estas redes no coincidían con las especificaciones de los servicios solicitados (maravilla de maravillas), por lo que se tuvieron que integrar subcapas convergentes en el modelo para proporcionar un espacio para documentar las diferencias. Por último, el comité esperaba en un principio que cada país tuviera una red, controlada por el gobierno y que utilizara los protocolos OSI, pero nunca pensaron en la interconectividad de redes. Para no hacer tan larga la historia, las cosas no sucedieron como se esperaba.

Con TCP/IP sucedió lo contrario: los protocolos llegaron primero y el modelo fue en realidad una descripción de los protocolos existentes. No había problemas para ajustar los protocolos al modelo. Encajaban a la perfección. El único problema era que el *modelo* no aceptaba otras pilas de protocolos. Como consecuencia, no era útil para describir otras redes que no fueran TCP/IP.

Volviendo de los asuntos filosóficos a los más específicos, una diferencia patente entre los dos modelos es el número de capas: el modelo OSI tiene siete y el TCP/IP sólo cuatro. Los dos tienen capas de (inter)red, transporte y aplicación, pero las otras capas son diferentes.

Otra diferencia está en el área de la comunicación orientada a la conexión comparada con la no orientada a la conexión. El modelo OSI soporta ambas comunicaciones en la capa de red, pero sólo la de comunicación orientada a la conexión en la capa de transporte, donde es importante (porque el servicio de transporte es transparente para los usuarios). El modelo TCP/IP sólo tiene un modo en la capa de red (no orientado a la conexión) pero soporta ambos modos en la capa de transporte, lo que da a los usuarios la oportunidad de elegir. Esta elección es importante especialmente para protocolos sencillos de solicitud-respuesta.

1.4.4 Crítica al modelo OSI y los protocolos

Ni el modelo OSI y sus protocolos ni el modelo TCP/IP y sus protocolos son perfectos. Se les pueden hacer, y se les han hecho, críticas. En ésta y en la siguiente sección veremos algunas de estas críticas. Empezaremos con el modelo OSI y más adelante examinaremos el modelo TCP/IP.

En la época en la que se publicó la segunda edición de este libro (1989), a muchos expertos en el campo les pareció que el modelo OSI y sus protocolos iban a dominar el mundo y a desplazar a todos los demás. Eso no sucedió. ¿Por qué? Sería útil echar un vistazo a algunas lecciones. Éstas se pueden resumir así:

1. Aparición inoportuna.
2. Mala tecnología.
3. Malas implementaciones.
4. Malas políticas.

Aparición inoportuna

Primero veamos la razón número uno: aparición inoportuna. El tiempo en que se establece un estándar es absolutamente crítico para el éxito. David Clark, del M.I.T., tiene una teoría de estándares que llama *apocalipsis de los dos elefantes,* la cual se ilustra en la figura 1-23.

Esta figura muestra la cantidad de actividad que rodea a un sujeto nuevo. Cuando se descubre primero el sujeto, hay una explosión de actividad de investigación en forma de exposiciones, documentos y reuniones. Después de un tiempo esta actividad disminuye, las empresas descubren el sujeto y surge la ola de miles de millones de dólares de inversión.

Es esencial que los estándares se escriban en el punto intermedio entre los dos "elefantes". Si los estándares se escriben demasiado pronto, antes de que se termine la investigación, el tema podría no estar entendido por completo; el resultado son malos estándares. Si se escriben demasiado tarde, varias empresas podrían haber hecho ya inversiones importantes en diversas maneras de hacer las cosas que los estándares han ignorado. Si el intervalo entre los dos elefantes es muy corto (porque cada cual tiene prisa por empezar), las personas que están desarrollando los estándares podrían fracasar.

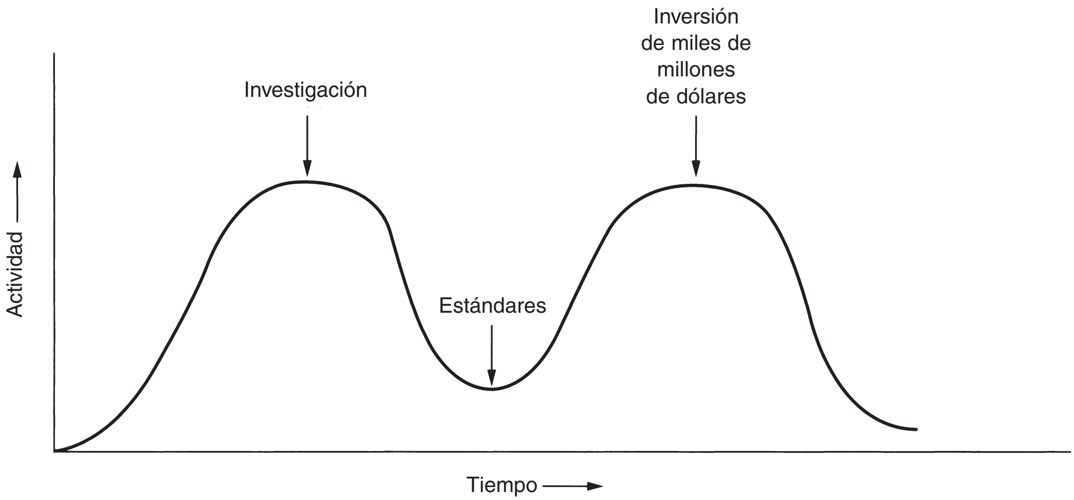


Figura 1-23. El apocalipsis de los dos elefantes.

Al parecer, los protocolos OSI estándar han sido vencidos. Los protocolos TCP/IP competidores ya eran ampliamente utilizados por las universidades investigadoras al momento en que aparecieron los protocolos OSI. Mientras la ola de los miles de millones de inversión aún no golpeaba, el mercado académico era bastante grande para que los proveedores empezaran a hacer ofertas cautas de los productos TCP/IP. Cuando OSI llegó, no quisieron soportar una segunda pila de protocolos hasta que se vieran forzados, por lo que no hubo ofertas iniciales. Puesto que cada empresa esperaba que la otra diera el primer paso, ninguna lo hizo y OSI nunca prosperó.

Mala tecnología

La segunda razón por la que OSI no tuvo éxito es que tanto el modelo como los protocolos tienen defectos. La elección de las siete capas fue más política que técnica, y dos de las capas (la de sesión y la de presentación) están casi vacías, mientras que las otras dos (la de enlace de datos y la de red) están saturadas.

El modelo OSI, junto con el servicio asociado de definiciones y protocolos, es extraordinariamente complejo. Si se apilan, los estándares impresos ocupan una fracción importante de un metro de papel. Incluso son difíciles de implementar y de operación deficiente. En este contexto, nos viene a la mente un enigma propuesto por Paul Mockapetris y citado en (Rose, 1993):

P: ¿Qué obtiene cuando cruza un gángster con un estándar internacional?
R: Alguien que le hace una oferta que usted no entiende.

Además de ser incomprensible, otro problema con OSI es que algunas funciones, como direccionamiento, control de flujo y control de errores, reaparecen una y otra vez en cada capa. Por

ejemplo, Saltzer y cols. (1984) han apuntado que para ser efectivo el control de errores, se debe hacer en la capa superior, puesto que repetirlo una y otra vez en cada una de las capas inferiores suele ser innecesario e ineficaz.

Malas implementaciones

Ante la enorme complejidad del modelo y los protocolos, no es de sorprender que las implementaciones iniciales fueran grandes, pesadas y lentas. Todos los que lo intentaron fracasaron. No le tomó mucho tiempo a las personas asociar OSI con "baja calidad". Aunque los productos mejoraron con el paso del tiempo, la imagen persistió.

En contraste, una de las primeras implementaciones de TCP/IP era parte de UNIX de Berkeley y fue bastante buena (sin mencionar que era gratis). Las personas pronto empezaron a utilizarla, lo que la llevó a un uso mayor por la comunidad, y esto a su vez condujo a mejoras que la llevaron a un mayor uso en la comunidad. Aquí la espiral fue ascendente en vez de descendente.

Malas políticas

A causa de la implementación inicial, muchas personas, sobre todo en el nivel académico, pensaban que TCP/IP era parte de UNIX, y en la década de 1980, UNIX no parecía tener paternidad alguna en la universidad.

Por otra parte, se tenía la idea de que OSI sería la criatura de los ministerios de telecomunicación de Europa, de la comunidad europea y más tarde del gobierno de los Estados Unidos. Esta creencia era cierta en parte, pero no ayudaba mucho la idea de un manojo de burócratas gubernamentales intentando poner en marcha un estándar técnicamente inferior al mando de los investigadores y programadores pobres que estaban en las trincheras desarrollando realmente redes de computadoras. Algunas personas compararon este desarrollo con la ocasión en que IBM anunció, en la década de 1960, que PL/I era el lenguaje del futuro, o cuando más tarde el DoD corregía esto anunciando que en realidad era Ada.

1.4.5 Crítica del modelo de referencia TCP/IP

El modelo de referencia TCP/IP y los protocolos también tienen sus problemas. En primer lugar, el modelo no distingue claramente los conceptos de servicio, interfaz y protocolo. Una buena ingeniería de software requiere la diferenciación entre la especificación y la implementación, algo que OSI hace con mucho cuidado y que TCP/IP no hace. En consecuencia, el modelo TCP/IP no es una guía para diseñar redes nuevas mediante tecnologías nuevas.

En segundo lugar, el modelo TCP/IP no es general del todo y no está bien ajustado para describir ninguna pila de protocolos más que de TCP/IP. Por ejemplo, es completamente imposible tratar de utilizar el modelo TCP/IP para describir Bluetooth.

En tercer lugar, la capa *host* a red no es en realidad una capa del todo en el sentido normal del término, como se utiliza en el contexto de los protocolos de capas. Es una interfaz (entre la capa de red y la de enlace de datos). La distinción entre una interfaz y una capa es crucial y nadie debe ser descuidado al respecto.

En cuarto lugar, el modelo TCP/IP no distingue (ni menciona) las capas física y de enlace de datos. Son completamente diferentes. La capa física tiene que ver con las características de transmisión de comunicación por cable de cobre, por fibra óptica e inalámbrica. El trabajo de la capa de enlace de datos es delimitar el inicio y fin de las tramas y captarlas de uno a otro lado con el grado deseado de confiabilidad. Un modelo adecuado debería incluir ambas como capas separadas. El modelo TCP/IP no hace esto.

Por último, aunque los protocolos IP y TCP se idearon e implementaron con sumo cuidado, muchos de los demás protocolos fueron hechos con fines específicos, producidos por lo general por estudiantes de licenciatura que los mejoraban hasta que se aburrían. Posteriormente, las implementaciones de tales protocolos se distribuyeron de manera gratuita, lo que dio como resultado un uso amplio y profundo y, por lo tanto, que fueran difíciles de reemplazar. Algunos de ellos ahora están en apuros. Por ejemplo, el protocolo de terminal virtual, TELNET, se diseñó para una terminal de teletipo mecánica de 10 caracteres por segundo. No sabe nada de interfaces gráficas de usuario ni de ratones. No obstante, 25 años más tarde aún tiene un amplio uso.

En resumen, a pesar de sus problemas, el *modelo* OSI (excepto las capas de sesión y presentación) ha probado ser excepcionalmente útil en la exposición de redes de computadoras. En contraste, los *protocolos* OSI no han sido muy populares. Sucede lo contrario con TCP/IP: el *modelo* es prácticamente inexistente, pero los *protocolos* tienen un amplio uso. En este libro utilizaremos un modelo OSI modificado pero nos concentraremos principalmente en el modelo TCP/IP y los protocolos relacionados, así como en los novísimos 802, SONET y Bluetooth. En efecto, utilizaremos el modelo híbrido de la figura 1-24 como marco de trabajo para este libro.

5	Capa de aplicación
4	Capa de transporte
3	Capa de red
2	Capa de enlace de datos
1	Capa física

Figura 1-24. Modelo de referencia híbrido que se usará en este libro.

1.5 REDES DE EJEMPLO

El tema de las redes de computadoras cubre muchos y diversos tipos de redes, grandes y pequeñas, bien conocidas y no tan bien conocidas. Tiene diferentes objetivos, escalamientos y tecnologías. En las siguientes secciones veremos algunos ejemplos para tener una idea de la variedad que se puede encontrar en el área de la conectividad de redes.

Empezaremos con Internet, que es probablemente la red más conocida y veremos su historia, evolución y tecnología. Luego consideraremos ATM, cuyo uso es frecuente en el núcleo de redes (telefónicas) grandes. Desde el punto de vista técnico difiere muy poco de Internet, y contrasta gratamente. Después presentaremos Ethernet, la red de área local dominante. Y, por último, veremos el IEEE 802.11, el estándar para las LANs inalámbricas.

1.5.1 Internet

Internet no es del todo una red, sino un inmenso conjunto de redes diferentes que usan ciertos protocolos comunes y proporcionan ciertos servicios comunes. Es un sistema poco común porque nadie lo planeó y nadie lo controla. Para entenderlo mejor, empecemos desde el principio y veamos cómo se desarrolló y por qué. Si desea leer una historia maravillosa sobre Internet, recomendamos ampliamente el libro de John Naughton (2000). Es uno de esos raros libros cuya lectura no sólo es divertida, sino que también contiene 20 páginas de *ibídems* y *op. cits*. para el historiador serio. Parte del material que se muestra a continuación se basa en dicho libro.

Desde luego, se ha escrito una infinidad de libros técnicos sobre Internet y sus protocolos. Para más información, vea (Maufer, 1999).

ARPANET

Nuestro relato empieza a fines de la década de 1950. Durante el auge de la Guerra Fría, el DoD quería una red de control y comando que pudiera sobrevivir a una guerra nuclear. En esa época todas las comunicaciones militares usaban la red telefónica pública, que se consideraba vulnerable. La razón de esta creencia se puede entresacar de la figura 1-25(a). Los puntos negros representan las oficinas de conmutación telefónica, a cada una de las cuales se conectaban miles de teléfonos. Estas oficinas de conmutación estaban, a su vez, conectadas a oficinas de conmutación de más alto nivel (oficinas interurbanas), para conformar una jerarquía nacional con sólo una mínima redundancia. La vulnerabilidad del sistema estaba en que la destrucción de algunas de las oficinas interurbanas clave podía fragmentar el sistema en muchas islas incomunicadas.

Hacia 1960, el DoD firmó un contrato con RAND Corporation para encontrar una solución. Uno de sus empleados, Paul Baran, presentó el diseño de amplia distribución y tolerancia a fallas que se muestra en la figura 1-25(b). Puesto que las trayectorias entre cualquiera de las oficinas de conmutación eran ahora más grandes de lo que las señales análogas podían viajar sin distorsión, Baran propuso que se utilizara la tecnología digital de conmutación de paquetes a través del sistema. Baran escribió varios informes al DoD describiendo en detalle sus ideas. A los oficiales del Pentágono les agradó el concepto y pidieron a AT&T, en ese entonces el monopolio telefónico estadounidense, que construyera un prototipo. AT&T desechó las ideas de Baran. La corporación más grande y rica del mundo no iba a permitir que un jovenzuelo le dijera cómo construir un sistema telefónico. Dijeron que la red de Baran no se podía construir y la idea se desechó.

Pasaron varios años y el DoD aún no tenía un mejor sistema de control y comando. Para entender qué sucedió a continuación, tenemos que volver al 7 de octubre de 1957, cuando la Unión soviética lanzó el Sputnik, su primer satélite artificial, con lo cual se le adelantó a Estados Unidos.

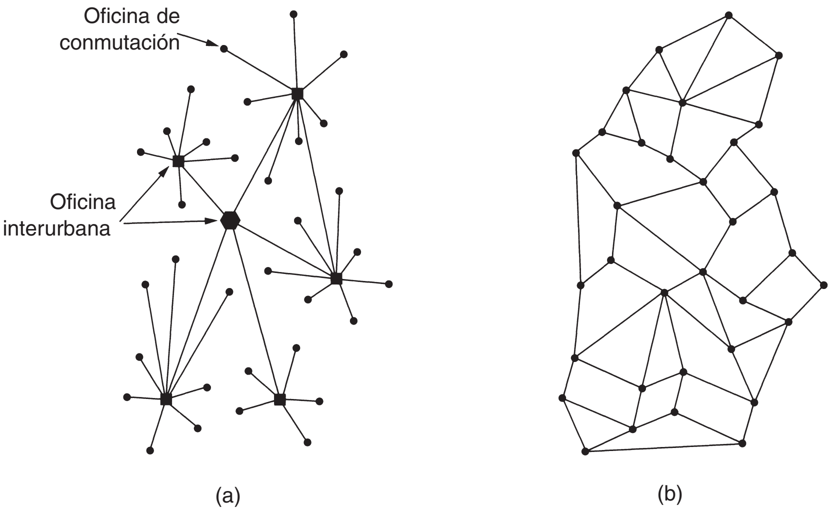


Figura 1-25. (a) Estructura de un sistema telefónico. (b) Sistema de conmutación distribuida propuesto por Baran.

Cuando el presidente Eisenhower trató de encontrar quién estaba dormido en sus laureles, se espantó al encontrarse con que la armada, el ejército y la fuerza aérea se peleaban por el presupuesto de investigación del Pentágono. Su respuesta inmediata fue crear una organización única de investigación para la defensa, **ARPA** (**Agencia de Proyectos de Investigación Avanzada**). Ésta no tenía científicos ni laboratorios; de hecho, no tenía más que una oficina y un presupuesto pequeño (por normas del Pentágono). Hacía su trabajo otorgando subvenciones y contratos a universidades y empresas cuyas ideas le parecían prometedoras.

Durante los primeros años, ARPA trataba de imaginarse cuál sería su misión, pero en 1967 la atención de su entonces director, Larry Roberts, se volvió hacia las redes. Se puso en contacto con varios expertos para decidir qué hacer. Uno de ellos, Wesley Clark, sugirió la construcción de una subred de conmutación de paquetes, dando a cada *host* su propio enrutador, como se ilustra en la figura 1-10.

Después del escepticismo inicial, Roberts aceptó la idea y presentó un documento algo vago al respecto en el Simposio sobre Principios de Sistemas Operativos ACM SIGOPS que se llevó a cabo en Gatlinburg, Tennessee, a fines de 1967 (Roberts, 1967). Para mayor sorpresa de Roberts, otro documento en la conferencia describía un sistema similar que no sólo había sido diseñado, sino que ya estaba implementado bajo la dirección de Donald Davies en el National Physical Laboratory en Inglaterra. El sistema del NPL no era un sistema a nivel nacional (sólo conectaba algunas computadoras en el campus del NPL), pero demostró que era posible hacer que la conmutación de paquetes funcionara. Además, citaba el trabajo anterior de Baran, el cual había sido descartado. Roberts salió de Gatlinburg determinado a construir lo que más tarde se conocería como **ARPANET.**

La subred constaría de minicomputadoras llamadas **IMPs (Procesadores de Mensajes de Interfaz),** conectadas por líneas de transmisión de 56 kbps. Para alta confiabilidad, cada IMP estaría conectado al menos a otros dos IMPs. La subred iba a ser de datagramas, de manera que si se destruían algunos IMPs, los mensajes se podrían volver a enrutar de manera automática a otras rutas alternativas.

Cada nodo de la red iba a constar de un IMP y un *host*, en el mismo cuarto, conectados por un cable corto. Un *host* tendría la capacidad de enviar mensajes de más de 8063 bits a su IMP, el cual los fragmentaría en paquetes de, a lo sumo, 1008 bits y los reenviaría de manera independiente hacia el destino. Cada paquete se recibiría íntegro antes de ser reenviado, por lo que la subred sería la primera red electrónica de conmutación de paquetes de almacenamiento y reenvío.

Entonces ARPA lanzó una convocatoria para construir la subred. Doce empresas licitaron. Después de evaluar las propuestas, ARPA seleccionó a BBN, una empresa de consultoría de Cambridge, Massachusetts, y en diciembre de 1968 le otorgó el contrato para construir la subred y escribir el software de ésta. BBN eligió utilizar como IMPs minicomputadoras Honeywell DDP-316 especialmente modificadas con palabras de 16 bits y 12 KB de memoria central. Los IMPs no tenían discos, ya que las partes móviles se consideraban no confiables. Estaban interconectadas por líneas de 56 kbps alquiladas a las compañías telefónicas. Aunque 56 kbps ahora es la elección de las personas que no pueden permitirse ADSL o cable, entonces era la mejor opción.

El software estaba dividido en dos partes: subred y *host*. El software de la subred constaba del extremo IMP de la conexión *host* a IMP, del protocolo IMP a IMP y de un protocolo de IMP origen a IMP destino diseñado para mejorar la confiabilidad. En la figura 1-26 se muestra el diseño original de ARPANET.

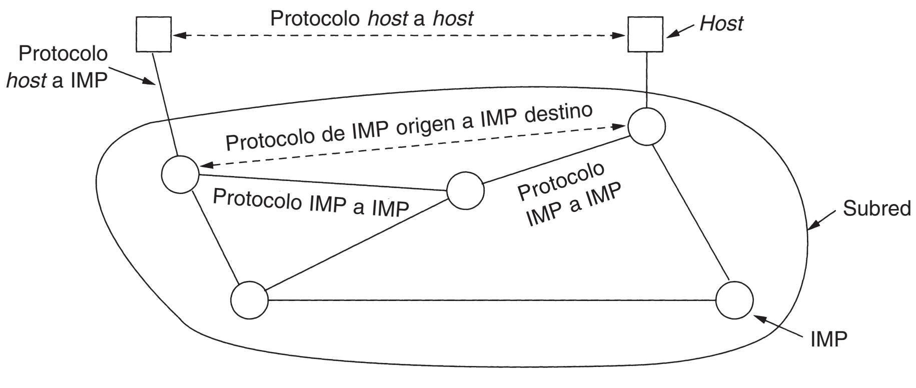


Figura 1-26. Diseño original de ARPANET.

Fuera de la subred también se necesitaba software, es decir, el extremo *host* de la conexión *host* a IMP, el protocolo *host* a *host* y el software de aplicación. Pronto quedó claro que BBN sintió que cuando se aceptaba un mensaje en un cable *host* a IMP y se ponía en un cable *host* a IMP en el destino, el trabajo estaba hecho.

Roberts tenía un problema: los *hosts* también necesitaban software. Para resolverlo convocó a una reunión de investigadores de red —en su mayoría estudiantes de licenciatura de Snowbird, Utah— durante el verano de 1969. Los estudiantes esperaban que algún experto en redes les explicara el gran diseño de la red y su software y que luego les asignara el trabajo de escribir parte de él. Se quedaron asombrados al descubrir que no había ningún experto ni un gran diseño. Tenían que averiguar qué era lo que se tenía que hacer.

No obstante, en diciembre de 1969 de alguna manera surgió una red experimental con cuatro nodos: en UCLA, UCSB, SRI y la Universidad de Utah. Se eligieron estas cuatro porque todas tenían un gran número de contratos de ARPA y todas tenían computadoras *host* diferentes incompatibles en su totalidad (precisamente para hacerlo más divertido). La red creció con rapidez a medida que se entregaban e instalaban más IMPs; pronto abarcó Estados Unidos. La figura 1-27 muestra qué tan rápido creció ARPANET en los primeros tres años.

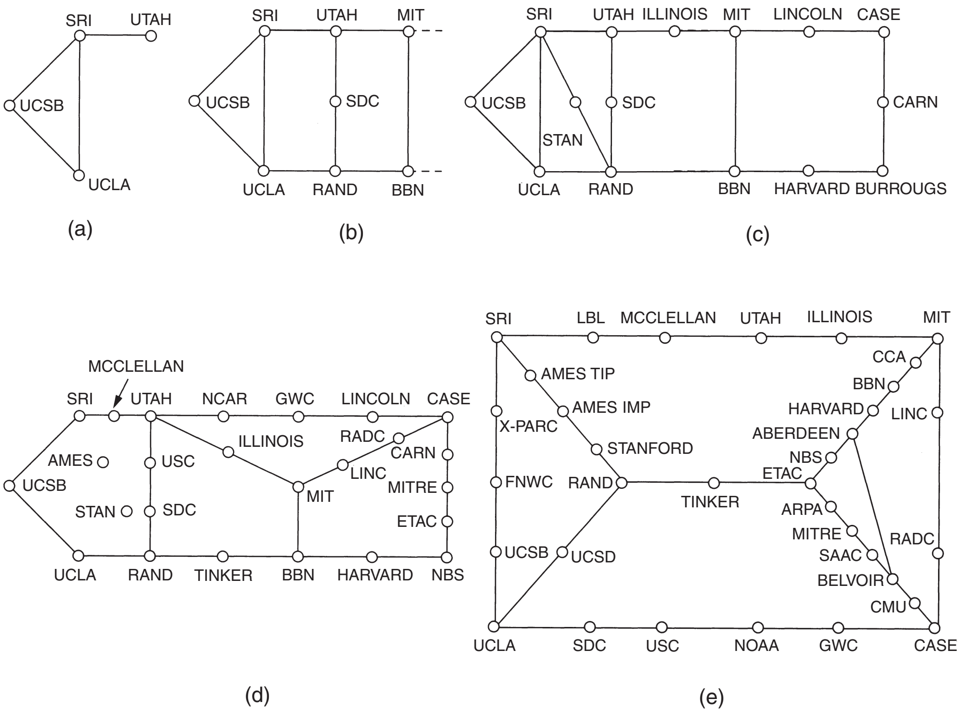


Figura 1-27. Crecimiento de ARPANET: (a) Diciembre de 1969. (b) Julio de 1970. (c) Marzo de 1971. (d) Abril de 1972. (e) Septiembre de 1972.

Además de ayudar al crecimiento de la novel ARPANET, ARPA también proporcionó fondos para la investigación sobre el uso de redes satelitales y redes de radio de paquetes móviles. En una demostración, ahora famosa, un camión que viajaba por California utilizó la red de radio de

paquetes para enviar mensajes a SRI, que luego los reenvió por ARPANET a la Costa Este, donde se expidieron al University College en Londres a través de una red satelital. Esto permitió que el investigador que iba en el camión usara una computadora que se encontraba en Londres mientras manejaba por California.

Este experimento también demostró que los protocolos existentes de ARPANET no eran adecuados para ejecutarse a través de varias redes. Esta observación condujo a más investigación sobre los protocolos, culminando con la invención del modelo y los protocolos de TCP/IP (Cerf y Kahn, 1974). TCP/IP está diseñado de manera específica para manejar comunicación por interredes, aspecto cuya importancia se acrecentó conforme cada vez más y más redes se adhirieron a ARPANET.

Para alentar la adopción de estos nuevos protocolos, ARPA concedió varios contratos a BBN y a la Universidad de California en Berkeley para integrarlos en UNIX de Berkeley. Los investigadores en Berkeley desarrollaron una interfaz de programa adecuada para la red (sockets) y escribieron muchos programas de aplicación, utilería y administración para hacer más fácil la conectividad.

El momento era perfecto. Muchas universidades habían adquirido recientemente una segunda o tercera computadora VAX y una LAN para conectarlas, pero no tenían software de redes. Cuando llegó 4.2BSD junto con TCP/IP, sockets y muchas utilerías de red, el paquete completo se adoptó de inmediato. Además, con TCP/IP, fue fácil para las LANs conectarse a ARPANET y muchas lo hicieron.

Durante la década de 1980, se conectaron redes adicionales, en particular LANs, a ARPANET. Conforme crecía el escalamiento, encontrar *hosts* llegó a ser muy costoso, por lo que se creó el **DNS** (**Sistema de Nombres de Dominio**) para organizar máquinas dentro de dominios y resolver nombres de *host* en direcciones IP. Desde entonces, el DNS ha llegado a ser un sistema de base de datos distribuido generalizado para almacenar una variedad de información relacionada con la elección de un nombre. En el capítulo 7 estudiaremos en detalle este tema.

NSFNET

A finales de la década de 1970, la NFS (Fundación Nacional para las Ciencias, de Estados Unidos) vio el enorme impacto que ARPANET estaba teniendo en la investigación universitaria, permitiendo que científicos de todo el país compartieran datos y colaboraran en proyectos de investigación. Sin embargo, para estar en ARPANET, una universidad debía tener un contrato de investigación con el DoD, lo cual muchas no tenían. La respuesta de la NSF fue diseñar un sucesor de ARPANET que pudiera estar abierto a todos los grupos de investigación de las universidades. Para tener algo concreto con que empezar, la NSF decidió construir una red dorsal (o troncal) para conectar sus seis centros de supercomputadoras en San Diego, Boulder, Champaign, Pittsburgh, Ithaca y Princeton. A cada supercomputadora se le dio un hermano menor, que consistía en una microcomputadora LSI-11 llamada ***fuzzball***. Estas computadoras estaban conectadas a líneas alquiladas de 56 kbps y formaban una subred, utilizando la misma tecnología de hardware que ARPANET. Sin embargo, la tecnología de software era diferente: las *fuzzball* utilizan TCP/IP desde el inicio, creando así la primera WAN TCP/IP.

La NSF también fundó algunas redes regionales (alrededor de 20) que se conectaban a la red dorsal para que los usuarios en miles de universidades, laboratorios de investigación, bibliotecas y museos, tuvieran acceso a cualquiera de las supercomputadoras y se comunicaran entre sí. Toda la red, incluyendo la red dorsal y las redes regionales, se llamó **NSFNET.** Ésta se conectó a ARPANET a través de un enlace entre un IMP y una *fuzzball* en el cuarto de máquinas de Carnegie-Mellon. En la figura 1-28 se muestra la primera red dorsal NSFNET.

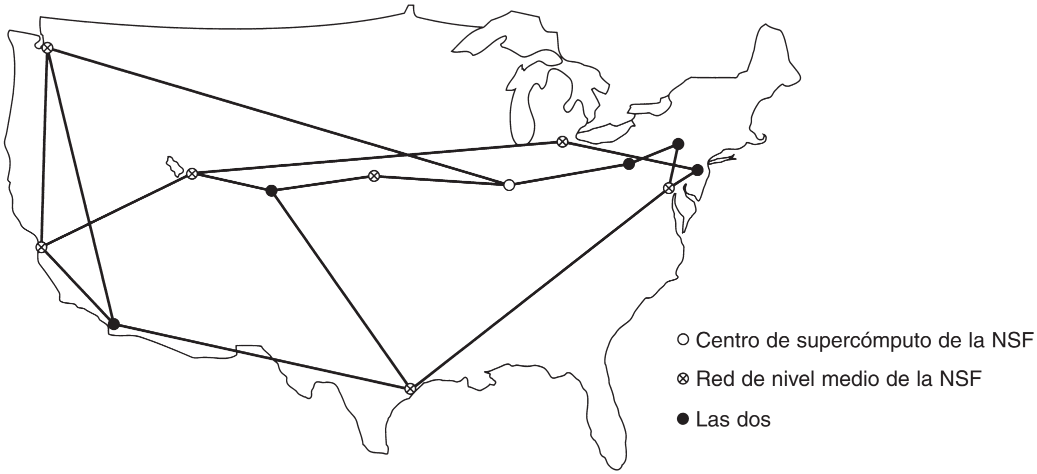


Figura 1-28. La red dorsal NSFNET en 1988.

NSFNET fue un éxito instantáneo y pronto se saturó. Inmediatamente, la NSF empezó a planear su sucesor y otorgó un contrato al consorcio MERIT de Michigan para que lo creara. Se alquilaron a MCI (puesto que se fusionó con WorldCom) canales de fibra óptica a 448 kbps para establecer la versión 2 de la red dorsal. Se utilizaron PC-RTs de IBM como enrutadores. Esta segunda red dorsal también se sobrecargó pronto, y en 1990 se escaló a 1.5 Mbps.

Al continuar el crecimiento, la NSF se percató de que el gobierno no podría financiar por siempre el uso de redes. Además, las empresas comerciales se querían unir, pero los estatutos de la NSF les prohibían utilizar las redes por las que la NSF había pagado. En consecuencia, la NSF alentó a MERIT, MCI e IBM a que formaran una corporación no lucrativa, **ANS** (**Redes y Servicios Avanzados**), como el primer paso hacia la comercialización. En 1990, ANS adquirió NSFNET y escaló los enlaces de 1.5 Mbps a 45 Mbps para formar **ANSNET.** Esta red operó durante cinco años y luego fue vendida a America Online. Pero para entonces varias empresas estaban ofreciendo servicios IP comerciales y fue evidente que el gobierno se debía retirar del negocio de las redes.

Para facilitar la transición y hacer que todas las redes regionales se pudieran comunicar con las demás redes regionales, la NSF concedió contratos a cuatro diferentes operadores de redes para establecer un **NAP** (**Punto de Acceso a la Red**). Estos operadores eran PacBell (San Francisco),

Ameritech (Chicago), MFS (Washington, D.C.) y Sprint (Nueva York, donde para efectos de NAP, Pennsauken, Nueva Jersey se toma en cuenta como si fuera la ciudad de Nueva York). Todo operador de red que quisiera proporcionar el servicio de red dorsal a las redes regionales de la NSF se tenía que conectar a todos los NAPs.

Este arreglo significaba que un paquete que se originara en cualquier red regional tenía la opción de contar con operadores de red dorsal desde su NAP al NAP de destino.

En consecuencia, los operadores de red dorsal se vieron forzados a competir por el negocio de las redes regionales con base en el servicio y el precio, que, desde luego, era la idea. Como resultado, el concepto de una única red dorsal predeterminada fue reemplazado por una infraestructura competitiva orientada a la comercialización. A muchas personas les gusta criticar al gobierno federal por no ser innovador, pero en el área de redes, el DoD y la NSF fueron los creadores de la infraestructura que cimentó la base para Internet y luego dejaron que la industria la operara.

Durante la década de 1990, muchos otros países y regiones también construyeron redes nacionales de investigación, con frecuencia siguiendo el patrón de ARPANET y NSFNET. Éstas incluían EuropaNET y EBONE en Europa, que empezaron con líneas de 2 Mbps y luego las escalaron a 34 Mbps. Finalmente, en Europa la infraestructura de redes quedó en manos de la industria.

Uso de Internet

El número de redes, máquinas y usuarios conectados a ARPANET creció rápidamente luego de que TCP/IP se convirtió en el protocolo oficial el 1o. de enero de 1983. Cuando NSFNET y ARPANET estaban interconectadas, el crecimiento se hizo exponencial. Muchas redes regionales se unieron y se hicieron conexiones a redes en Canadá, Europa y el Pacífico.

En algún momento a mediados de la década de 1980, las personas empezaron a ver el conjunto de redes como una interred y más tarde como Internet, aunque no hubo una inauguración oficial con algún político rompiendo una botella de champaña sobre una *fuzzball*.

El aglutinante que mantiene unida la Internet es el modelo de referencia TCP/IP y la pila de protocolos de TCP/IP. TCP/IP hace posible el servicio universal y se puede comparar con la adopción de la medida estándar para el ancho de vía del ferrocarril en el siglo XIX o la adopción de los protocolos de señalización comunes para las compañías telefónicas.

¿Qué significa en realidad estar en Internet? Nuestra definición es que una máquina está en Internet si ejecuta la pila de protocolos de TCP/IP, tiene una dirección IP y puede enviar paquetes IP a todas las demás máquinas en Internet. La sola capacidad para enviar y recibir correo electrónico no basta, puesto que el correo electrónico es la puerta de entrada a muchas redes fuera de Internet. Sin embargo, el aspecto se nubla de alguna manera por el hecho de que millones de computadoras personales pueden llamar a un proveedor de servicios de Internet mediante un módem, recibir direcciones IP temporales y enviar paquetes IP a otros *hosts* de Internet. Tiene sentido decir que tales máquinas están en Internet en tanto estén conectadas al enrutador del proveedor de servicios.

Tradicionalmente (es decir, de 1970 a 1990) Internet y sus predecesores tenían cuatro aplicaciones principales:

1. **Correo electrónico.** La capacidad para redactar, enviar y recibir correo electrónico ha sido posible desde los inicios de ARPANET y su gran popularidad. Muchas personas obtienen docenas de mensajes al día y consideran esto como su primer medio de interactuar con el mundo exterior, más allá del teléfono y el correo caracol que se han quedado atrás. Hoy en día los programas de correo electrónico están disponibles en prácticamente todo tipo de computadora.

2. **Noticias.** Los grupos de noticias son foros especializados en los que los usuarios con un interés común pueden intercambiar mensajes. Existen miles de grupos de noticias, dedicados a temas técnicos y no técnicos, entre ellos computadoras, ciencia, recreación y política. Cada grupo de noticias tiene su propia etiqueta, estilo, hábitos y penas en que se incurre al violarlas.

3. **Inicio remoto de sesión.** Mediante los programas telnet, rlogin o ssh, los usuarios de cualquier parte en Internet pueden iniciar sesión en cualquier otra máquina en la que tengan una cuenta.

4. **Transferencia de archivos.** Con el programa FTP, los usuarios pueden copiar archivos de una máquina en Internet a otra. Por este medio se encuentra disponible una vasta cantidad de artículos, bases de datos y otra información.

Hasta principios de la década de 1990, Internet era muy visitada por investigadores académicos, del gobierno e industriales. Una nueva aplicación, **WWW** (**World Wide Web**) cambió todo eso y trajo millones de usuarios nuevos no académicos a la red. Esta aplicación —inventada por Tim Berners-Lee, físico del CERN— no cambió ninguna de las características subyacentes pero las hizo más fáciles de usar. Junto con el navegador Mosaic, escrito por Marc Andreessen en el Centro Nacional para Aplicaciones de Supercómputo en Urbana, Illinois, WWW hizo posible que un sitio estableciera páginas de información que contienen texto, imágenes, sonido e incluso vídeo, y vínculos integrados a otras páginas. Al hacer clic en un vínculo, el usuario es transportado de inmediato a la página a la que apunta dicho vínculo. Por ejemplo, muchas compañías tienen una página de inicio con entradas que apuntan a otras páginas que contienen información de productos, listas de precios, ventas, soporte técnico, comunicación con empleados, información para los accionistas y más.

En muy poco tiempo han aparecido páginas de otro tipo, incluyendo mapas, tablas del mercado accionario, catálogos de fichas bibliográficas, programas de radio grabados e incluso una página que apunta al texto completo de muchos libros cuyos derechos de autor han expirado (Mark Twain, Charles Dickens, etcétera). Muchas personas también tienen páginas personales (páginas de inicio).

Gran parte de su crecimiento durante la década de 1990 estuvo alimentado por empresas llamadas **ISPs** (**proveedores de servicios de Internet**). Hay compañías que ofrecen a los usuarios individuales domésticos la capacidad de llamar a una de sus máquinas y conectarse a Internet, obteniendo así acceso al correo electrónico, WWW y otros servicios de Internet. Estas compañías suscribieron contratos con decenas de millones de usuarios nuevos por un año durante el final de

la década de 1990, cambiando por completo el carácter de la red de ser un campo de recreo para académicos y militares a uno de utilidad pública, muy semejante al sistema telefónico. Ahora el número de usuarios de Internet se desconoce, pero lo cierto es que son cientos de millones en todo el mundo y tal vez pronto lleguen a rebasar los mil millones.

Arquitectura de Internet

En esta sección trataremos de dar un breve panorama de lo que es Internet hoy. Debido a las muchas fusiones entre compañías telefónicas (telcos) e ISPs, las aguas se han enturbiado y a veces es difícil decir quién hace qué. En consecuencia, la siguiente descripción será, por necesidad, algo más sencilla que la realidad. En la figura 1-29 se muestra el panorama general. Ahora examinemos esta figura parte por parte.

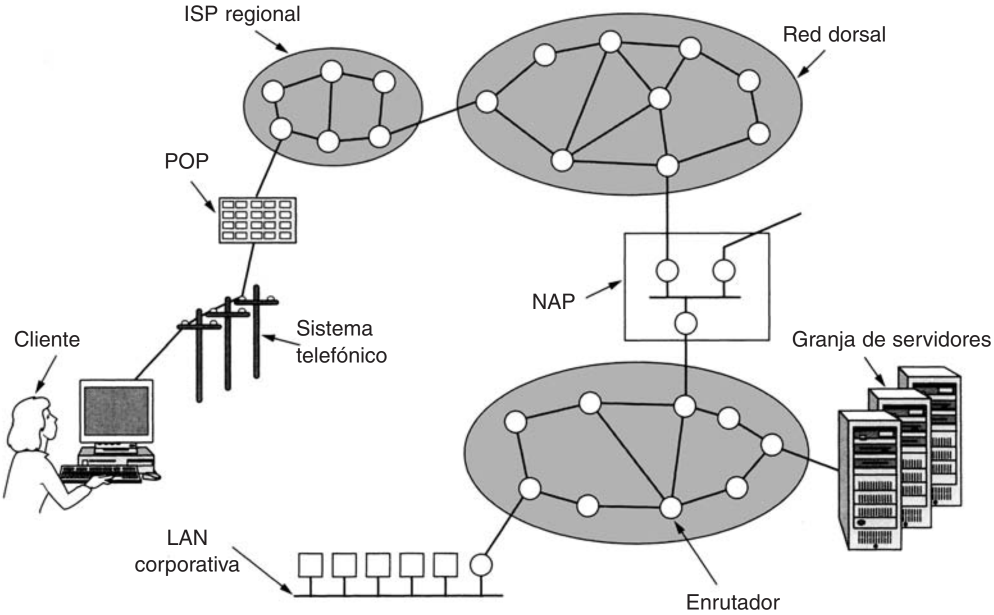


Figura 1-29. Panorama de Internet.

Un buen lugar para empezar es con un cliente en casa. Supongamos que nuestro cliente llama a su ISP desde una conexión de línea telefónica conmutada, como se muestra en la figura 1-29. El módem es una tarjeta dentro de su PC que convierte las señales digitales que la computadora produce en señales análogas que pueden pasar sin obstáculos a través del sistema telefónico. Estas señales se transfieren al **POP** (**Punto de Presencia**) del ISP, donde se retiran del sistema telefónico y se inyectan en la red regional del ISP. A partir de este punto, el sistema es totalmente digital y de conmutación de paquetes. Si el ISP es la telco local, es probable que el POP esté ubicado en la

oficina de conmutación telefónica, donde termina el cableado de teléfono de los clientes. Si el ISP no es la telco local, el POP podría ser alguna de las oficinas de conmutación en el camino.

La red regional de ISPs consta de enrutadores interconectados en las diversas ciudades en las que el ISP opera o da servicio. Si el paquete está destinado a un *host* servido directamente por el ISP, el paquete se entrega al *host*. En caso contrario, se entrega al operador de la red dorsal del ISP.

En la cima de la cadena alimenticia están los operadores de las principales redes dorsales, empresas como AT&T y Sprint. Éstas operan grandes redes de redes dorsales internacionales, con miles de enrutadores conectados por fibra óptica de banda ancha. Grandes corporaciones y servicios de *hosting* que ejecutan granjas de servidores (máquinas que pueden servir miles de páginas Web por segundo) con frecuencia se conectan de manera directa a la red dorsal. Los operadores de redes dorsales alientan esta conexión directa rentando espacio en lo que se llama **hoteles de portadores,** que son básicamente gabinetes de equipos en el mismo cuarto que el enrutador para conexiones cortas y rápidas entre las granjas de servidores y la red dorsal.

Si un paquete dado a la red dorsal se destina a un ISP o a una compañía servida por la red dorsal, se envía al enrutador más cercano y se pierde cualquier responsabilidad por este paquete. Sin embargo, en el mundo hay muchas redes dorsales, de varios tamaños, de manera que un paquete podría tener que ir a una red dorsal competidora. Para que los paquetes viajen entre redes dorsales, todas las redes principales se conectan a los NAPs explicados antes. Básicamente, un NAP es un cuarto lleno de enrutadores, al menos uno por red dorsal. Una LAN en el cuarto conecta todos los enrutadores, de modo que los paquetes se pueden reenviar desde una red dorsal hacia cualquier otra. Además de estar conectadas en los NAPs, las redes dorsales más grandes tienen numerosas conexiones directas entre sus enrutadores, una técnica conocida como **igualdad privada** (*private peering*). Una de las muchas paradojas de Internet es que los ISPs que compiten en público entre sí por clientes, con frecuencia cooperan estableciendo igualdades privadas entre ellos (Metz, 2001).

Aquí termina nuestro rápido viaje por Internet. En los siguientes capítulos tendremos mucho que decir sobre los componentes individuales y su diseño, algoritmos y protocolos. También vale la pena mencionar de paso que algunas empresas tienen interconectadas todas sus redes internas existentes, utilizando con frecuencia la misma tecnología que Internet. Por lo general, estas **intranets** son accesibles sólo dentro de la empresa pero, por otra parte, funcionan del mismo modo que Internet.

1.5.2 Redes orientadas a la conexión: X.25, Frame Relay y ATM

Desde el inicio de la conectividad surgió una guerra entre aquellos que apoyan a las subredes no orientadas a la conexión (es decir, de datagramas) y quienes apoyan a las subredes orientadas a la conexión. Los principales defensores de las subredes no orientadas a la conexión vienen de la comunidad ARPANET/Internet. Recuerde que el deseo original del DoD al fundar y construir ARPANET era tener una red que pudiera funcionar incluso después de que varios impactos de armas nucleares destruyeran numerosos enrutadores y líneas de transmisión. Por lo tanto, la tolerancia a

errores era importante en su lista de prioridades, no tanto lo que pudieran cobrar a los clientes. Este enfoque condujo a un diseño no orientado a la conexión en el que cada paquete se enruta independientemente de cualquier otro paquete. Por lo tanto, si algunos enrutadores se caen durante una sesión, no hay daño puesto que el sistema puede reconfigurarse a sí mismo de manera dinámica para que los paquetes subsiguientes puedan encontrar alguna ruta a su destino, aun cuando sea diferente de la que utilizaron los paquetes anteriores.

El campo orientado a la conexión viene del mundo de las compañías telefónicas. En el sistema telefónico, quien llama debe marcar el número de la parte a la que desea llamar y esperar la conexión antes de poder hablar o enviar los datos. Esta configuración de conexión establece una ruta a través del sistema telefónico que se mantiene hasta que se termina la llamada. Todas las palabras o paquetes siguen la misma ruta. Si una línea o conmutador se cae en el trayecto, la llamada se cancela. Esta propiedad es precisamente lo que al DoD no le gustaba.

Entonces, ¿por qué le gustaba a las compañías telefónicas? Por dos razones:

1. Calidad en el servicio.
2. Facturación.

Al establecer de antemano una conexión, la subred puede reservar recursos como espacio de búfer y capacidad de procesamiento (CPU) en los enrutadores. Si se intenta establecer una llamada y los recursos disponibles son insuficientes, la llamada se rechaza y el invocador recibe una señal de ocupado. De esta manera, una vez que se establece una conexión, ésta da un buen servicio. Con una red no orientada a la conexión, si llegan demasiados paquetes al mismo enrutador al mismo tiempo, el enrutador se saturará y tal vez pierda algunos paquetes. Tal vez el emisor advierta esto y los envíe de nuevo, pero la calidad del servicio será accidentada e inadecuada para audio o vídeo a menos que la red tenga poca carga. No es necesario decir que proveer una calidad de audio adecuada es algo en lo que las compañías telefónicas ponen mucho cuidado, de ahí su preferencia por las conexiones.

La segunda razón por la que las compañías telefónicas prefieren el servicio orientado a la conexión es que están acostumbradas a cobrar por el tiempo de conexión. Cuando hace una llamada de larga distancia (sea nacional o internacional) se le cobra por minuto. Cuando llegaron las redes, se vieron atraídas precisamente hacia un modelo en el que el cobro por minuto fuera fácil de hacer. Si se tiene que establecer una conexión antes de enviar los datos, en ese momento es cuando el reloj de la facturación empieza a correr. Si no hay conexión, no hay cobro.

Irónicamente, mantener registros de facturación es muy costoso. Si una compañía telefónica adoptara una tarifa mensual plana sin límite de llamadas y sin facturación o mantenimiento de un registro, probablemente ahorraría una gran cantidad de dinero, a pesar del incremento en llamadas que generaría esta política. Sin embargo, hay políticas, regulaciones y otros factores que pesan en contra de hacer esto. Curiosamente, el servicio de tarifa plana existe en otros sectores. Por ejemplo, la TV por cable se factura en una tasa mensual plana, independientemente de cuántos programas vea. Podría haberse diseñado con pago por evento como concepto básico, pero no fue así, en parte por lo costoso de la facturación (y dada la calidad de la mayoría de los programas televisivos, la vergüenza no se puede descontar del todo). Asimismo, muchos parques de diversiones

cobran una cuota de admisión por día con acceso ilimitado a los juegos, en contraste con las ferias ambulantes que cobran por juego.

Dicho esto, no nos debería sorprender que todas las redes diseñadas por la industria de la telefonía hayan sido subredes orientadas a la conexión. Lo que sí es de sorprender es que Internet también se está inclinado en esa dirección, a fin de proporcionar un mejor servicio de audio y vídeo, un tema al que volveremos en el capítulo 5. Por ahora examinaremos algunas redes orientadas a la conexión.

X.25 y Frame Relay

Nuestro primer ejemplo de red orientada a la conexión es la **X.25,** que fue la primera red de datos pública. Se desplegó en la década de 1970, cuando el servicio telefónico era un monopolio en todas partes y la compañía telefónica de cada país esperaba que hubiera una red de datos por país —la propia. Para utilizar X.25, una computadora establecía primero una conexión con la computadora remota, es decir, hacía una llamada telefónica. Esta conexión daba un número de conexión para utilizarlo en los paquetes de transferencia de datos (ya que se podían abrir muchas conexiones al mismo tiempo). Los paquetes de datos eran muy sencillos, consistían en un encabezado de 3 bytes y hasta 128 bytes de datos. El encabezado constaba de un número de conexión de 12 bits, un número de secuencia de paquete, un número de confirmación de recepción y algunos bits diversos. Las redes X.25 funcionaron por casi diez años con resultados mixtos.

En la década de 1980, las redes X.25 fueron reemplazadas ampliamente por un nuevo tipo de red llamada **Frame Relay.** Ésta es una red orientada a la conexión sin controles de error ni de flujo. Como era orientada a la conexión, los paquetes se entregaban en orden (en caso de que se entregaran todos). Las propiedades de entrega en orden, sin control de errores ni de flujo hicieron el Frame Relay parecido a la LAN de área amplia. Su aplicación más importante es la interconexión de LANs en múltiples oficinas de una empresa. Frame Relay disfrutó de un éxito modesto y aún se sigue utilizando en algunas partes.

Modo de Transferencia Asíncrona

Otro tipo de red orientada a la conexión, tal vez el más importante, es **ATM** (**Modo de Transferencia Asíncrona**)**.** La razón de tan extraño nombre se debe a que en el sistema telefónico la mayor parte de la transmisión es síncrona (lo más parecido a un reloj), y en ATM no sucede así.

ATM se diseñó a principios de la década de 1990 y se lanzó en medio de una increíble exageración (Ginsburg, 1996; Goralski, 1995; Ibe, 1997; Kimn y cols., 1994, y Stallings, 2000). ATM iba a resolver todos los problemas de conectividad y telecomunicaciones fusionando voz, datos, televisión por cable, télex, telégrafo, palomas mensajeras, botes conectados por cordón, tambores, señales de humo y todo lo demás, en un solo sistema integrado que pudiera proporcionar todos los servicios para todas las necesidades. Eso no sucedió. En gran parte, los problemas fueron semejantes a los ya descritos en el tema de OSI, es decir, una aparición inoportuna, junto con tecnología, implementación y políticas equivocadas. Habiendo noqueado a las compañías telefónicas en el primer asalto, gran parte de la comunidad de Internet vio a ATM como cuando Internet era el contrincante de las telcos: la segunda parte. Pero no fue así en realidad y esta vez incluso los in-

transigentes fanáticos de los datagramas se dieron cuenta de que la calidad de servicio de Internet dejaba mucho que desear. Para no alargar la historia, ATM tuvo mucho más éxito que OSI y actualmente tiene un uso profundo dentro del sistema telefónico, con frecuencia en el transporte de los paquetes IP. Como en la actualidad las empresas portadoras la utilizan principalmente para su transporte interno, los usuarios no se percatan de su existencia pero, definitivamente, vive y goza de salud.

Circuitos virtuales de ATM

Puesto que las redes ATM están orientadas a la conexión, el envío de datos requiere que primero se envíe un paquete para establecer la conexión. Conforme el mensaje de establecimiento sigue su camino a través de la subred, todos los conmutadores que se encuentran en la ruta crean una entrada en sus tablas internas tomando nota de la existencia de la conexión y reservando cualesquier recursos que necesite la conexión. Con frecuencia a las conexiones se les conoce como **circuitos virtuales,** en analogía con los circuitos físicos utilizados en el sistema telefónico. La mayoría de las redes ATM soportan también **circuitos virtuales permanentes,** que son conexiones permanentes entre dos *hosts* (distantes). Son similares a las líneas alquiladas del mundo telefónico. Cada conexión, temporal o permanente, tiene un solo identificador de conexión. En la figura 1-30 se ilustra un circuito virtual.

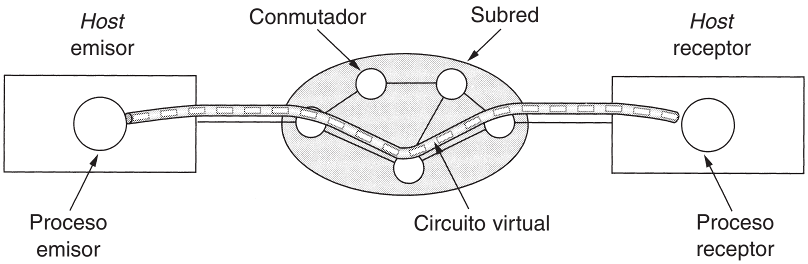


Figura 1-30. Un circuito virtual.

Una vez establecida la conexión, cada lado puede empezar a transmitir datos. La idea básica en que se fundamenta ATM es transmitir toda la información en paquetes pequeños, de tamaño fijo, llamados **celdas.** Las celdas tienen un tamaño de 53 bytes, de los cuales cinco son del encabezado y 48 de carga útil, como se muestra en la figura 1-31. Parte del encabezado es el identificador de la conexión, por lo que los *hosts* emisor y receptor y todos los conmutadores intermedios pueden saber qué celdas pertenecen a qué conexiones. Esta información permite que cada conmutador sepa cómo enviar cada celda entrante. La conmutación de celdas se hace en el hardware, a alta velocidad. De hecho, el principal argumento para tener celdas de tamaño fijo es que así es fácil construir conmutadores de hardware para manejar celdas pequeñas, de longitud fija. Los paquetes de longitud variable de IP se tienen que enrutar mediante software, que es un proceso más lento.

Otro punto a favor de ATM es que el hardware se puede configurar para enviar una celda entrante a múltiples líneas de salida, una propiedad necesaria para el manejo de un programa de televisión que se va a difundir a varios receptores. Por último, las celdas pequeñas no bloquean ninguna línea por mucho tiempo, lo que facilita la garantía en la calidad del servicio.

Todas las celdas siguen la misma ruta al destino. La entrega de celdas no está garantizada, pero el orden sí. Si las celdas 1 y 2 se envían en ese orden, entonces deben arribar en el mismo orden, nunca primero la 2 y luego la 1. No obstante, una de las dos o ambas se pueden perder en el trayecto. A los niveles más altos de protocolos les corresponde la recuperación de celdas perdidas. Observe que aunque esta garantía no es perfecta, es mejor que la de Internet. Ahí los paquetes no sólo se pierden, sino que además se entregan en desorden. ATM, en contraste, garantiza que las celdas nunca se entregarán en desorden.

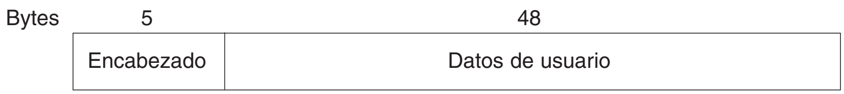


Figura 1-31. Una celda ATM.

Las redes ATM se organizan como las WANs tradicionales, con líneas y conmutadores (enrutadores). Las velocidades más comunes para las redes ATM son de 155 y 622 Mbps, aunque también se soportan velocidades más altas. Se eligió la velocidad de 155 Mbps porque ésta es la que se requiere para transmitir televisión de alta definición. La elección exacta de 155.52 Mbps se hizo por compatibilidad con el sistema de transmisión SONET de AT&T, punto que estudiaremos en el capítulo 2. La velocidad de 622 Mbps se eligió para que se pudieran enviar cuatro canales de 155 Mbps.

El modelo de referencia ATM

ATM tiene su propio modelo de referencia, el cual es diferente del OSI y también del TCP/IP. En la figura 1.32 se muestra el modelo de referencia ATM. Consta de tres capas: la física, la ATM y la de adaptación ATM, además de lo que el usuario desee poner arriba de ellas.

La capa física tiene que ver con el medio físico: voltajes, temporización de bits y otros aspectos más. ATM no prescribe un conjunto particular de reglas, tan sólo especifica que las celdas ATM se pueden enviar tal cual por cable o fibra, pero también se pueden empacar dentro de la carga útil de otros sistemas de transporte. En otras palabras, ATM se ha diseñado para ser independiente del medio de transmisión.

La **capa ATM** se encarga de las celdas y su transporte. Define la disposición de una celda e indica qué significan los campos del encabezado. También tiene que ver con el establecimiento y la liberación de los circuitos virtuales. El control de congestión también se ubica aquí.

Puesto que la mayoría de las aplicaciones no necesita trabajar de manera directa con las celdas (aunque algunas podrían hacerlo), se ha definido una capa superior a la capa ATM para que

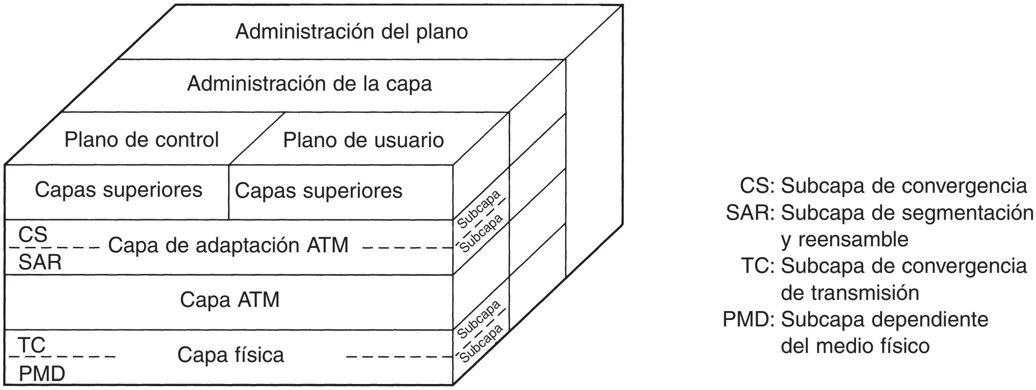


Figura 1-32. El modelo de referencia ATM.

los usuarios envíen paquetes más grandes que una celda. La interfaz de ATM segmenta estos paquetes, transmite de forma individual las celdas y las reensambla en el otro extremo. Esta capa es la **AAL** (**Capa de Adaptación ATM**).

A diferencia de los primeros modelos de referencia bidimensionales, el modelo ATM se define como si fuera tridimensional, lo que se puede apreciar en la figura 1-32. El **plano de usuario** trata con el transporte de datos, control de flujo, corrección de errores y otras funciones de usuario. En contraste, el **plano de control** se ocupa de la administración de la conexión. Las funciones de administración del plano y de la capa se relacionan con la administración de recursos y coordinación entre capas.

Cada una de las capas física y AAL se dividen en dos subredes, una en la parte inferior que hace el trabajo y en la subcapa de convergencia en la parte superior que proporciona la interfaz propia de la capa superior inmediata. En la figura 1-33 se muestran las funciones de las capas y subcapas.

La subcapa **PMD** (**Dependiente del Medio Físico**) interactúa con el cable real. Mueve los bits dentro y fuera y maneja la temporización de bits, es decir, el tiempo que existe entre cada bit al transmitirlos. Esta capa será diferente para diferentes transportadoras y cables.

La otra subcapa de la capa física es la subcapa **TC** (**Convergencia de Transmisión**). Cuando se transmiten las celdas, la capa TC las envía como una cadena de bits a la capa PMD. Esto es sencillo. En el otro extremo, la subcapa TC recibe una serie de bits de entrada de la subcapa PMD. Su trabajo es convertir este flujo de bits en un flujo de celdas para la capa ATM. Maneja todos los aspectos relacionados con las indicaciones de dónde empiezan y terminan las celdas en el flujo de bits. En el modelo ATM, esta funcionalidad se da en la capa física. En el modelo OSI y en gran parte de las demás redes, el trabajo de entramado, es decir, convertir una serie de bits en bruto en una secuencia de tramas o celdas, es la tarea de la capa de enlace de datos.

Como mencionamos antes, la capa ATM maneja celdas, incluyendo su generación y transporte. La mayoría de los aspectos interesantes de ATM se encuentra ubicada aquí. Es una combinación de las capas de enlace de datos y de red del modelo OSI; no hay una división en subcapas.

Capa OSI	Capa ATM	Subcapa ATM	Funcionalidad
3/4	AAL	CS	Provisión de la interfaz estándar (convergencia)
		SAR	Segmentación y reensamblado
2/3	ATM		Control de flujo Generación/extracción de encabezado de celda Circuito virtual/administración de ruta Multiplexión/desmultiplexión de celdas
2	Física	TC	Desacoplamiento de proporción de celdas Generación y comprobación de la suma de verificación de encabezados Generación de celdas Empaque/desempaque de celdas a partir del sobre contenedor Generación de tramas
1		PMD	Temporización de bits Acceso a la red física

Figura 1-33. Las capas y subcapas de ATM y sus funciones.

La capa AAL se divide en una subcapa **SAR** (**Segmentación y Reensamble**) y una **CS** (**Subcapa de Convergencia**)**.** La subcapa inferior fragmenta paquetes en celdas en el lado de transmisión y los une de nuevo en el destino. La subcapa superior permite que los sistemas ATM ofrezcan diversos tipos de servicios a diferentes aplicaciones (por ejemplo, la transferencia de archivos y el vídeo bajo demanda tienen diferentes requerimientos respecto a manejo de errores, temporización, etcétera).

Puesto que quizá ATM esté en declive, no lo explicaremos más en este libro. No obstante, puesto que existe una base instalada considerable, es probable que aún siga en uso durante algunos años. Para más información sobre ATM, vea (Dobrowsky y Grise, 2001, y Gadecki y Heckart, 1997).

1.5.3 Ethernet

Internet y ATM se diseñaron para conectividad de área amplia. Sin embargo, muchas empresas, universidades y otras organizaciones tienen un gran número de computadoras que requieren interconexión. Esta necesidad dio origen a la red de área local. En esta sección diremos algo sobre la LAN más popular: Ethernet.

La historia empieza en la prístina Hawaii a principios de la década de 1970. En este caso, “prístina” se puede interpretar como “que no tiene un sistema telefónico funcional”. En tanto los días son más agradables para los vacacionistas cuando no son interrumpidos por el teléfono, no fue así para el investigador Norman Abramson y sus colegas de la Universidad de Hawaii, quienes estuvieron tratando de conectar usuarios de las islas remotas a la computadora principal de Hono-

lulu. Conectar sus propios cables bajo el Océano Pacífico parecía imposible, de modo que buscaron una solución diferente.

La primera que encontraron fueron los radios de onda corta. Cada terminal estaba equipada con un radio pequeño de dos frecuencias: un canal ascendente (a la computadora central) y otro descendente (desde la computadora central). Cuando el usuario deseaba conectarse con la computadora, sólo transmitía por el canal ascendente un paquete que contenía los datos. Si en ese instante nadie más estaba transmitiendo, probablemente el paquete saldría y su recepción sería confirmada en el canal descendente. Si había contención por el canal ascendente, la terminal detectaría la falta de confirmación de recepción y haría otro intento. Puesto que sólo habría un emisor en el canal descendente (la computadora central), nunca habría colisiones ahí. Este sistema, llamado ALOHANET, trabajaba muy bien en condiciones de bajo tráfico pero se caía cuando el flujo de tráfico ascendente era pesado.

En esa misma época, un estudiante llamado Bob Metcalfe hizo su licenciatura en el M.I.T. y luego se mudó para obtener su doctorado en Harvard. Durante sus estudios, conoció el trabajo de Abramson. Se interesó tanto en él que después de graduarse en Harvard decidió pasar el verano en Hawaii trabajando con Abramson antes de empezar a trabajar en el Centro de Investigación de Palo Alto de Xerox (PARC). Cuando llegó al PARC, vio que los investigadores de ahí habían diseñado y construido lo que más tarde se llamarían computadoras personales. Pero las máquinas estaban aisladas. Aplicando su conocimiento del trabajo de Abramson, junto con su colega David Boggs, diseñó e implementó la primera red de área local (Metcalfe y Boggs, 1976).

Llamaron **Ethernet** al sistema, por lo de *luminiferous ether,* a través del cual se pensó alguna vez que se propagaba la radiación electromagnética. (Cuando, en el siglo XIX, el físico inglés James Clerk Maxwell descubrió que la radiación electromagnética se podía describir mediante una ecuación de onda, los científicos supusieron que el espacio debía estar lleno de algún medio etéreo en el cual se propagaba la radiación. Sólo después del famoso experimento de Michelson-Morley en 1887, los físicos descubrieron que la radiación electromagnética se podía propagar por el vacío.)

Aquí el medio de transmisión no era el vacío, sino un cable coaxial grueso (el éter) de más de 2.5 km de largo (con repetidoras cada 500 metros). El sistema podía contener hasta 256 máquinas por medio de transceptores acoplados al cable. Un cable con múltiples máquinas en paralelo se llama **cable de derivación múltiple** (*multidrop*). El sistema se ejecutaba a 2.94 Mbps. En la figura 1-34 se presenta un esbozo de su arquitectura. Ethernet tenía una mejora importante respecto de ALOHANET; antes de transmitir, una computadora tenía que escuchar el cable para ver si había alguien más transmitiendo. En caso de que ya lo hubiera, la computadora se mantenía en espera de que la transmisión actual terminara. Al hacerlo así se evitaba interferir con las transmisiones existentes, dando una mayor eficiencia. ALOHANET no trabajaba de este modo porque para una terminal en una isla era imposible detectar la transmisión de otra terminal en una isla distante. El problema se resolvía con un cable único.

A pesar de que la computadora escucha antes de transmitir, surge un problema: ¿qué sucede si dos o más computadoras esperan hasta que se complete la transmisión actual y luego empiezan

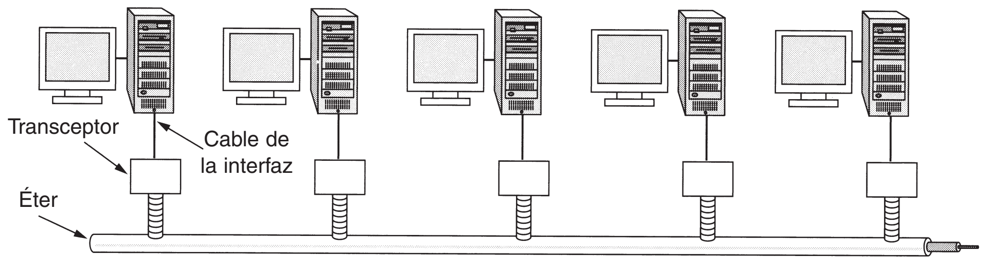


Figura 1-34. Arquitectura de la Ethernet original.

a transmitir al mismo tiempo? La solución es que cada computadora escuche durante su propia transmisión y, si detecta interferencia, mande una señal para poner en alerta a todos los transmisores. Después espera un tiempo aleatorio antes de reintentarlo. Si sucede una colisión, el tiempo aleatorio de espera se duplica y así sucesivamente, para separar las transmisiones que están en competencia y dar a alguna la oportunidad de transmitir primero.

La Ethernet de Xerox fue tan exitosa que DEC, Intel y Xerox diseñaron un estándar en 1978 para una Ethernet de 10 Mbps, llamado **estándar DIX.** Con dos cambios menores, en 1983 el estándar DIX se convirtió en el estándar IEEE 802.3.

Por desgracia para Xerox, ya tenía fama de hacer inventos originales (como el de las computadoras personales) y luego fallar en la comercialización de los mismos, como se menciona en un relato titulado *Fumbling the Future* (Smith y Alexander, 1988). Cuando Xerox mostró poco interés en hacer algo con Ethernet aparte de ayudar a estandarizarlo, Metcalfe formó su propia empresa, 3Com, con el propósito de vender adaptadores Ethernet para PCs. Ha vendido más de 100 millones.

Ethernet continuó su desarrollo y aún está en desarrollo. Han salido nuevas versiones a 100 y 1000 Mbps, e incluso más altas. También se ha mejorado el cableado y se han agregado conmutación y otras características. En el capítulo 4 explicaremos Ethernet en detalle.

De paso, vale la pena mencionar que Ethernet (IEEE 802.3) no es el único estándar de LAN. El comité también estandarizó Token Bus (802.4) y Token Ring (802.5). La necesidad de tres estándares más o menos incompatibles tiene poco que ver con la tecnología y mucho con la política. En el momento de la estandarización, General Motors estaba impulsando una LAN en la que la topología era la misma que la usada en Ethernet (un cable linear), pero las computadoras transmitían por turnos pasando un pequeño paquete de computadora a computadora, llamado **token.** Una computadora podía transmitir sólo si poseía el *token*, lo que evitaba colisiones. General Motors anunció que este esquema era esencial para la manufactura de automóviles y que no estaba preparado para cambiar su postura. No obstante este anuncio, el 802.4 prácticamente desapareció.

Del mismo modo, IBM tenía su favorito: su Token Ring patentado. En este esquema el *token* se pasaba a través del anillo y la computadora que poseyera el *token* podía transmitir antes de poner el *token* de nuevo en el anillo. A diferencia del 802.4, este esquema, estandarizado como 802.5,

aún se usa en algunos sitios de IBM, pero prácticamente en ninguna parte más. Sin embargo, se está desarrollando una versión de 1 gigabit (802.5v), pero parece poco probable que alcance a Ethernet. Resumiendo, había una guerra entre Ethernet, Token Bus y Token Ring, pero Ethernet ganó, en gran medida porque fue la primera y los retadores no pudieron superarlo.

1.5.4 LANs inalámbricas: 802.11

Casi al mismo tiempo que aparecieron las computadoras portátiles, muchas personas tuvieron el sueño de andar por la oficina y poder conectar a Internet su computadora. En consecuencia, varios grupos empezaron a trabajar para cumplir con esta meta. El método más práctico es equipar las computadoras de la oficina y las portátiles con transmisores y receptores de radio de onda corta que les permitan comunicarse. Este trabajo condujo rápidamente a que varias empresas empezaran a comercializar las LANs inalámbricas.

El problema es que no había compatibilidad entre ninguna de ellas. Esta proliferación de estándares implicaba que una computadora equipada con un radio de marca *X* no funcionara en un cuarto equipado con una estación de base marca *Y*. Finalmente, la industria decidió que un estándar de LAN inalámbrica sería una buena idea, por lo que al comité del IEEE que estandarizó las LANs alámbricas se le encargó la tarea de diseñar un estándar para LANs inalámbricas. El estándar resultante se llamó 802.11. En la jerga común se le conoce como **WiFi.** Es un estándar importante y merece respeto, así que lo llamaremos por su nombre propio, 802.11.

El estándar propuesto tenía que trabajar en dos modos:

1. En presencia de una estación base.
2. En ausencia de una estación base.

En el primer caso, toda la comunicación se hacía a través de la estación base, que en la terminología del 802.11 se conoce como **punto de acceso.** En el segundo caso, las computadoras podrían enviarse mensajes entre sí directamente. Este modo se llama a veces **red ad hoc.** Un ejemplo típico es el de dos o más personas que se encuentran juntas en un cuarto no equipado con una LAN inalámbrica y cuyas computadoras se comunican entre sí de manera directa. Los dos modos se ilustran en la figura 1-35.

La primera decisión fue la más sencilla: cómo llamarlo. Todos las otros estándares LAN tenían números como 802.1, 802.2, hasta 802.10, por lo que el estándar LAN se llamó o publicó como 802.11. El resto fue más difícil.

En particular, varios de los diversos retos que había que enfrentar eran: encontrar una banda de frecuencia adecuada, de preferencia mundial; enfrentar el hecho de que las señales de radio tienen un rango finito; asegurarse de que se mantuviera la privacidad de los usuarios; tomar en cuenta la vida limitada de las baterías; preocuparse por la seguridad humana (¿las ondas de radio causan cáncer?); comprender las implicaciones de la movilidad de las computadoras y, por último, construir un sistema con suficiente ancho de banda para que sea económicamente viable.

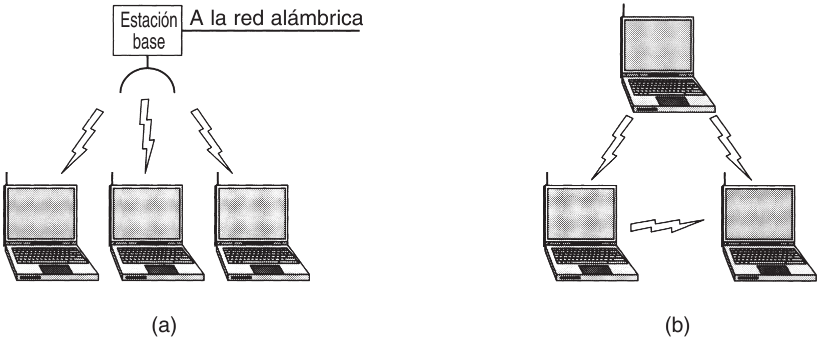


Figura 1-35. (a) Red inalámbrica con una estación base. (b) Red *ad hoc*.

Cuando empezó el proceso de estandarización (a mediados de la década de 1990), Ethernet ya había llegado a dominar las redes de área local, por lo que el comité decidió hacer que el 802.11 fuera compatible con Ethernet sobre la capa de enlace de datos. En particular, se podría enviar un paquete IP sobre la LAN inalámbrica del mismo modo en que una computadora conectada mediante cable enviaba un paquete IP a través de Ethernet. No obstante, existen algunas diferencias inherentes con Ethernet en las capas física y de enlace de datos y tuvieron que manejarse mediante el estándar.

Primero, una computadora en Ethernet siempre escucha el medio antes de transmitir. Sólo si el medio está inactivo la computadora puede empezar a transmitir. Esta idea no funciona igual en las LANs inalámbricas. Para ver por qué, examine la figura 1-36. Suponga que la computadora *A* está transmitiendo a la computadora *B,* pero el alcance del radio del transmisor de *A* es muy corto para encontrar a la computadora *C*. Si *C* desea transmitir a *B* puede escuchar el medio antes de empezar, pero el hecho de que no escuche nada no quiere decir que su transmisión tendrá éxito. El estándar 802.11 tenía que resolver este problema.

El segundo problema que se tenía que resolver es que los objetos sólidos pueden reflejar una señal de radio, por lo que ésta se podría recibir múltiples veces (a través de varias rutas). Esta interferencia da como resultado lo que se llama **desvanecimiento por múltiples trayectorias.**

El tercer problema es que una gran cantidad de software no toma en cuenta la movilidad. Por ejemplo, muchos procesadores de texto tienen una lista de impresoras de entre las cuales los usuarios pueden elegir para imprimir un archivo. Cuando la computadora en la que se ejecuta el procesador de texto se coloca en un nuevo entorno, la lista interna de impresoras ya no es útil.

El cuarto problema es que si una computadora portátil se mueve lejos de la estación base que está usando y dentro del rango de una estación base diferente, se requiere algún tipo de manejo. Aunque este problema ocurre con los teléfonos celulares, eso no sucede con Ethernet y requiere solución. En particular, la red prevista consta de múltiples celdas, cada una con su propia estación base pero con las estaciones base conectadas por Ethernet, como se muestra en la figura 1-37. Desde fuera todo el sistema se vería como una Ethernet sola. La conexión entre el sistema 802.11 y el mundo exterior se conoce como **portal.**

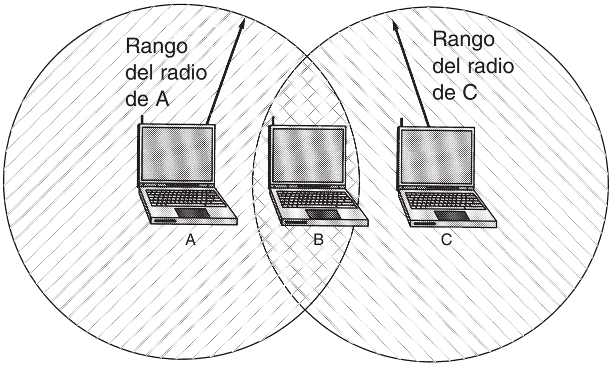


Figura 1-36. El rango de un solo radio no podría cubrir todo el sistema.

Después de algún trabajo, el comité se presentó en 1997 con un estándar que se dirigía a éstos y otros respectos. La LAN inalámbrica descrita se ejecutaba a 1 o 2 Mbps. Casi de inmediato la gente comenzó a quejarse de que era demasiado lenta, de manera que empezaron a trabajar en estándares más rápidos. Una división desarrollada con el comité tuvo como resultado dos nuevos estándares en 1999. El estándar 802.11a utiliza una banda de frecuencia más ancha y se ejecuta a velocidades de hasta 54 Mbps. El estándar 802.11b utiliza la misma banda de frecuencia que el 802.11, pero se vale de una técnica de modulación diferente para alcanzar 11 Mbps. Algunas personas ven esto como un aspecto psicológico importante puesto que 11 Mbps es más rápido que la Ethernet alámbrica original. Es posible que el 802.11 original de 1 Mbps desaparezca con rapidez, pero aún no queda claro cuál de los nuevos estándares será el ganador.

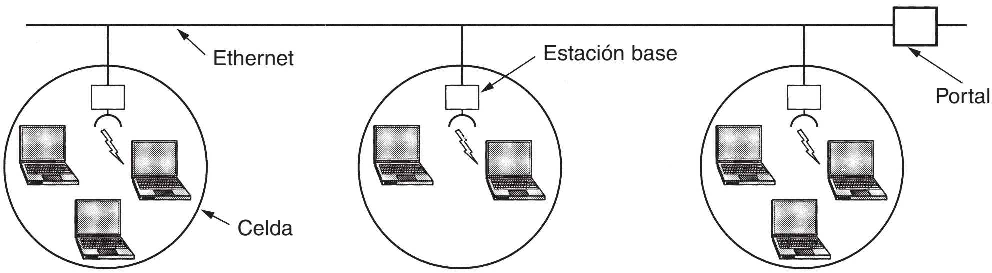


Figura 1-37. Una red 802.11 de múltiples celdas.

Para hacer las cosas todavía más complicadas, el comité 802 ha creado otra variante, el 802.11g, que utiliza la técnica de modulación del 802.11a pero la banda de frecuencia del 802.11b. En el capítulo 4 trataremos en detalle al 802.11.

Sin lugar a dudas, el 802.11 va a causar una revolución en computación y en el acceso a Internet. Aeropuertos, estaciones de trenes, hoteles, centros comerciales y universidades lo están instalando rápidamente. Incluso cafeterías de lujo están instalando el 802.11 para que los *yuppies* que se reúnen puedan navegar en Web mientras toman su café con leche. Es posible que el 802.11 haga por Internet lo que las computadoras portátiles hicieron por la computación: volverla móvil.

1.6 ESTANDARIZACIÓN DE REDES

Existen muchos fabricantes y proveedores de redes, cada uno con sus propias ideas de cómo se deben hacer las cosas. Sin coordinación sería un caos total y los usuarios nunca conseguirían nada. La única manera de resolver esta situación es ponerse de acuerdo en la adopción de algunos estándares para redes.

Los estándares no sólo permiten que computadoras diferentes se comuniquen, sino que también incrementan el mercado de productos que se ajustan al estándar. Un mercado grande conduce a la producción masiva, economías de escala en la producción, implementaciones VLSI y otros beneficios que disminuyen el precio e incrementan aún más la aceptación. En las siguientes secciones daremos un vistazo al importante, pero poco conocido, mundo de la estandarización internacional.

Los estándares se dividen en dos categorías: de facto y de jure. Los estándares ***de facto*** ("de hecho") son los que simplemente surgieron, sin ningún plan formal. La PC de IBM y sus sucesoras son estándares de facto para oficinas chicas y equipos domésticos porque docenas de fabricantes decidieron copiar exactamente las máquinas de IBM. Del mismo modo, UNIX es el estándar de facto para sistemas operativos en los departamentos de ciencias de la computación de las universidades.

En contraste, los estándares ***de jure*** ("por derecho"), son formales, legales, adoptados por alguna institución de estandarización autorizada. Por lo general, las autoridades de estandarización internacional se dividen en dos clases: las establecidas por acuerdos entre los gobiernos de cada país, y las incluidas de manera voluntaria, sin acuerdos entre organizaciones. En el área de los estándares de redes de computadoras hay varias organizaciones de cada tipo, las cuales explicaremos a continuación.

1.6.1 Quién es quién en el mundo de las telecomunicaciones

La situación legal de las compañías telefónicas del mundo varía considerablemente de un país a otro. En un extremo están los Estados Unidos con sus 1500 empresas telefónicas privadas individuales. Antes de que AT&T se dividiera, en 1984, era la empresa más grande del mundo y dominaba el escenario. Proporcionaba servicio telefónico a casi 80% de los usuarios de Estados

Unidos, con sucursales diseminadas en la mitad del país; las compañías oponentes atendían al resto de los clientes (en su mayor parte rurales). Desde que se dividió, AT&T sigue proporcionando servicio de larga distancia, pero ahora en competencia con otras empresas. Las siete Compañías Operadoras Regionales de Bell que surgieron de la división de AT&T y numerosas empresas independientes proporcionan servicios de telefonía local y celular. Debido a las frecuentes fusiones y otros cambios, la industria está en un estado de movimiento constante.

Las empresas que dan servicios de comunicación al público en Estados Unidos se llaman **portadoras comunes** (*carriers*). Las ofertas y precios se describen en un documento llamado **tarifa,** el cual debe ser aprobado por la Comisión Federal de Comunicaciones para el tráfico interestatal e internacional, pero el tráfico estatal interno lo aprueban las comisiones de servicios públicos.

En el otro extremo están los países en los cuales el gobierno respectivo tiene el monopolio de todas las comunicaciones, como correos, telégrafos, teléfonos y a veces la radio y la televisión. La mayor parte del mundo cae dentro de esta categoría. En algunos casos la autoridad de la telecomunicación es una compañía nacionalizada y en otros es simplemente una rama del gobierno, conocida generalmente como **PTT** (administración de **Correos, Telégrafos y Teléfonos**)**.** La tendencia a nivel mundial es hacia una liberación y competencia, y alejarse del monopolio gubernamental. La mayoría de los países europeos tiene privatizadas (parcialmente) sus PTTs, pero en otras partes el proceso avanza con lentitud.

Con tantos proveedores diferentes de servicios, es claro que se necesita una compatibilidad a escala mundial para asegurarse de que las personas (y las computadoras) de un país puedan llamar a sus contrapartes en otro. En realidad, esta necesidad ha existido desde hace mucho tiempo. En 1865, los representantes de muchos gobiernos de Europa se reunieron para formar el predecesor de la actual **ITU** (**Unión Internacional de Telecomunicaciones**)**.** Su trabajo era estandarizar las telecomunicaciones internacionales, que en esos días se hacían mediante el telégrafo. Incluso entonces era patente que si la mitad de los países utilizaba el código Morse y la otra utilizaba un código diferente, surgiría un problema. Cuando el teléfono entró al servicio internacional, la ITU empezó a trabajar en la estandarización de la telefonía. En 1947 la ITU se convirtió en una agencia de las Naciones Unidas.

La ITU tiene tres sectores principales:

1. Radiocomunicaciones (ITU-R).
2. Estandarización de telecomunicaciones (ITU-T).
3. Desarrollo (ITU-D).

La ITU-R se ocupa de asignar frecuencias de radio en todo el mundo a los grupos de interés en competencia. Nos enfocaremos en primer lugar en la ITU-T, que se ocupa de los sistemas telefónicos y de comunicación de datos. De 1956 a 1993, la ITU-T se conocía como **CCITT** (del francés *Comité Consultatif International Télégraphique et Téléphonique,* Comité Consultivo Internacional para la Telegrafía y Telefonía). El 1o. de marzo de 1993 el CCITT se reorganizó para hacerlo menos burocrático y cambió de nombre para reflejar su nuevo papel. Tanto la ITU-T como el CCITT emitieron recomendaciones en el área de comunicaciones telefónicas y de datos.

Es frecuente encontrar algunas de las recomendaciones del CCITT, como la X.25 del CCITT, aunque desde 1993 las recomendaciones llevan la etiqueta de la ITU-T.

La ITU-T tiene cuatro clases de miembros:

1. Gobiernos nacionales.
2. De sector.
3. Asociados.
4. Agencias reguladoras.

La ITU-T tiene alrededor de 200 miembros gubernamentales, entre ellos casi todos los miembros de las Naciones Unidas. Puesto que Estados Unidos no tiene una PTT, alguien más tenía que representarlos en la ITU-T. Esta tarea recayó en el Departamento de Estado, probablemente porque la ITU-T tenía que ver con los países extranjeros, que era la especialidad del Departamento de Estado.

Hay aproximadamente 500 miembros de sector, incluyendo compañías telefónicas (por ejemplo, AT&T, Vodafone, WorldCom), fabricantes de equipos de telecomunicación (como Cisco, Nokia, Nortel), fabricantes de computadoras (como Compaq, Sun, Toshiba), fabricantes de chips (como Intel, Motorola, TI), compañía de medios (como AOL Time Warner, CBS, Sony) y otras empresas interesadas (como Boeing, Samsung, Xerox). Varias organizaciones científicas no lucrativas y consorcios industriales también son miembros de sector (por ejemplo, IFIP e IATA). Los miembros asociados son organizaciones más pequeñas que se interesan en un grupo de estudio en particular. Las agencias reguladoras son quienes vigilan el negocio de la telecomunicación, como la Comisión Federal de Comunicaciones de Estados Unidos.

La tarea de la ITU-T es hacer recomendaciones técnicas sobre telefonía, telegrafía y las interfaces de comunicación de datos. Estas recomendaciones suelen convertirse en estándares reconocidos internacionalmente, por ejemplo el V.24 (también conocido en Estados Unidos como EIA RS-232), el cual especifica la ubicación y significado de los diversos pines en el conector utilizado para la mayoría de las terminales asíncronas y módems externos.

Es preciso observar que las recomendaciones de la ITU-T técnicamente son sólo sugerencias que los gobiernos pueden adoptar o ignorar (ya que los gobiernos parecen adolescentes de 13 años a quienes no les gusta recibir órdenes). En la práctica, un país que desee adoptar un estándar telefónico diferente del utilizado por el resto del mundo, es libre de hacerlo, pero el precio es el aislamiento. Esto podría funcionar en Corea del Norte, pero fuera de ahí sería un verdadero problema. El sofisma de llamar "recomendaciones" a los estándares de la ITU-T era y es necesario para mantener en calma el nacionalismo de varios países.

El trabajo verdadero de la ITU-T se realiza en sus 14 grupos de estudio, a veces de hasta 400 personas, que abarcan aspectos que van desde la facturación telefónica hasta servicios de multimedia. Para conseguir la realización de los proyectos, los grupos de estudio se dividen en equipos de trabajo, que a su vez se dividen en equipos de expertos, que a su vez se dividen en grupos específicos. Una vez burócrata, jamás se deja de serlo.

A pesar de todo esto, en realidad la ITU-T hace su trabajo. Desde que surgió, ha producido cerca de 3000 recomendaciones que ocupan cerca de 60,000 páginas de papel. Muchas de ellas se han llevado a la práctica en gran medida. Por ejemplo, una de sus recomendaciones es el popular estándar V.90 para módems de 56 kbps.

En tanto las comunicaciones completen la transición, que empezó en la década de 1980, de ser nacionales totalmente a ser globales totalmente, los estándares llegarán a ser más importantes cada vez, y más y más organizaciones querrán estar implicadas en su establecimiento. Para más información sobre la ITU, vea (Irmer, 1994).

1.6.2 Quién es quién en los estándares internacionales

Los estándares internacionales son producidos y publicados por la **ISO** (**Organización de Estándares Internacionales**),† una organización voluntaria no surgida de un acuerdo, fundada en 1946. Sus miembros son las organizaciones de estándares nacionales de los 89 países miembro. Entre ellos se encuentran ANSI (Estados Unidos), BSI (Gran Bretaña), AFNOR (Francia), DIN (Alemania) y otros 85.

La ISO emite estándares sobre una gran cantidad de temas, desde los más básicos (literalmente) como tuercas y pernos, hasta el revestimiento de los postes telefónicos (sin mencionar las semillas de cacao [ISO 2451], las redes de pesca [ISO 1530], ropa interior femenina [ISO 4416] y algunos otros objetos que no se pensaría que fueran sujetos de estandarización). Se han emitido más de 13,000 estándares, entre ellos los estándares de OSI. La ISO tiene casi 200 comités técnicos, numerados por el orden de su creación, refiriéndose cada uno a un objeto específico. El TC1 se ocupa de las tuercas y pernos (estandariza la rosca de los tornillos). El TC97 trata con computadoras y procesamiento de información. Cada TC tiene subcomités (SCs) divididos en grupos de trabajo (WGs).

El trabajo real lo hacen sobre todo los WGs, integrados por más de 100,000 voluntarios en todo el mundo. Muchos de estos "voluntarios" son asignados a trabajar en asuntos de la ISO por sus empleadores, cuyos productos se están estandarizando. Otros son oficiales gubernamentales ansiosos de que lo que se hace en su país llegue a ser estándar internacional. Los expertos académicos también están activos en muchos de los WGs.

En cuanto a estándares de telecomunicación, la ISO y la ITU-T suelen cooperar (la ISO es miembro de la ITU-T), para evitar la ironía de dos estándares internacionales oficiales mutuamente incompatibles.

El representante de Estados Unidos en la ISO es el **ANSI** (**Instituto Estadounidense de Estándares Nacionales**), que a pesar de su nombre es una organización privada no gubernamental y no lucrativa. Sus miembros son fabricantes, empresas portadoras comunes y otras partes interesadas. La ISO suele adoptar los estándares ANSI como estándares internacionales.

El procedimiento seguido por la ISO para adoptar estándares se ha diseñado para obtener el mayor consenso posible. El proceso inicia cuando alguna de las organizaciones de estándares

†Para los puristas, el verdadero nombre de la ISO es Organización Internacional para la Estandarización.

nacionales siente la necesidad de un estándar internacional en un área determinada. Entonces se forma un grupo de trabajo para presentar un **CD** (**Borrador de Comité**)**.** El CD se distribuye a todos los miembros, que tienen seis meses para criticarlo. Si la mayoría lo aprueba, se revisa y distribuye un documento revisado, llamado **DIS** (**Borrador de Estándar Internacional**) para comentarios y votación. Con base en los resultados de esta vuelta, se prepara, aprueba y publica el texto final del **IS** (**Estándar Internacional**)**.** En áreas de gran controversia, un CD o un DIS podría llegar a tener varias versiones antes de lograr suficientes votos y todo el proceso puede llegar a tardar años.

El **NIST** (**Instituto Nacional de Estándares y Tecnología**) es parte del Departamento de Comercio de Estados Unidos. Se llamaba Oficina Nacional de Estándares. Emite estándares que son obligatorios para compras hechas por el gobierno de Estados Unidos, excepto por los del Departamento de Defensa, que tiene sus propios estándares.

Otro representante importante en el mundo de los estándares es el **IEEE** (**Instituto de Ingenieros Eléctricos y Electrónicos**)**,** la mayor organización de profesionales del mundo. Además de publicar multitud de periódicos y organizar cientos de conferencias cada año, el IEEE tiene un grupo de estandarización que desarrolla estándares en el área de ingeniería eléctrica y computación. El comité 802 del IEEE ha estandarizado muchos tipos de LANs. Estudiaremos algunos de sus resultados más adelante. El trabajo real lo hace un conjunto de grupos de trabajo, que se listan en la figura 1.38. La tasa de éxito de los diversos grupos de trabajo del 802 ha sido baja; el hecho de tener un número 802.x no garantiza el éxito. Pero el impacto de las historias de éxito (en especial, del 802.3 y el 802.11) ha sido tremendo.

1.6.3 Quién es quién en el mundo de los estándares de Internet

El amplio mundo de Internet tiene sus propios mecanismos de estandarización, muy diferentes de los de la ITU-T y la ISO. La diferencia se puede resumir diciendo que quienes asisten a las reuniones de estandarización de la ITU o la ISO van de traje, pero las personas que asisten a las juntas de estandarización de Internet van de mezclilla (excepto cuando se reúnen en San Diego, donde van de *short* y camiseta).

En las reuniones de la ITU y la ISO abundan los oficiales corporativos y burócratas, para quienes la estandarización es su trabajo. Se refieren a la estandarización como una Cosa Buena y dedican sus vidas a ella. Por otro lado, la gente de Internet prefiere la anarquía por cuestión de principios. Sin embargo, con cientos de millones de personas haciendo sus propias cosas, la comunicación es escasa. Por lo tanto, los estándares, aunque deplorables, son necesarios.

Cuando se configuró ARPANET, el DoD creó un comité informal para supervisarla. En 1983 se dio otro nombre al comité: **IAB** (**Consejo de Actividades de Internet**) y se le encomendó una misión un poco más amplia, que era la de mantener a los investigadores de ARPANET y de Internet apuntando más o menos en la misma dirección; algo muy parecido a juntar una manada de gatos. El significado del acrónimo “IAB” se cambió a **Consejo para la Arquitectura de Internet.**

Número	Tema
802.1	Supervisión y arquitectura de LANs
802.2 ↓	Control lógico de enlace
802.3 *	Ethernet
802.4 ↓	Token bus (se utilizó por un corto tiempo en plantas manufactureras)
802.5	Token ring (entrada de IBM al mundo de las LANs)
802.6 ↓	Cola dual, bus dual (primera red de área metropolitana)
802.7 ↓	Grupo de consultoría técnico de tecnologías de banda ancha
802.8 †	Grupo de consultoría de tecnologías de fibra óptica
802.9 ↓	LANs síncrona (para aplicaciones de tiempo real)
802.10 ↓	LANs virtuales y seguridad
802.11 *	LANs inalámbricas
802.12 ↓	Demanda de prioridad (AnyLAN de Hewlett-Packard)
802.13	Número de mala suerte. Nadie lo quiso
802.14 ↓	Módems de cable (desaparecido: primero surgió un consorcio en la industria)
802.15 *	Redes de área personal (Bluetooth)
802.16 *	Redes inalámbricas de área ancha
802.17	Anillo de paquete elástico

Figura 1-38. Los grupos de trabajo del 802. Los importantes se marcan con *. Los que se marcan con ↓ están en hibernación. El que tiene la † se desintegró.

Cada uno de los aproximadamente 10 miembros del IAB encabezaba una fuerza de trabajo relacionada con algún asunto importante. El IAB se reunía varias veces al año para discutir los resultados y para dar retroalimentación al DoD y a la NSF, que proporcionaban la mayor parte de los fondos en aquel entonces. Cuando se necesitaba un estándar (por ejemplo, un nuevo algoritmo de enrutamiento), los miembros del IAB le daban solución y después anunciaban los cambios para que los estudiantes que estuvieran a cargo de la implementación del software pudieran realizarlos. La comunicación se llevaba a cabo mediante una serie de informes técnicos denominados **RFCs** (**Solicitudes de Comentarios).** Estos informes se almacenan en línea y cualquiera que esté interesado en ellos puede descargarlos de *www.ietf.org/rfc.* Los RFCs se encuentran organizados por el orden cronológico de su creación. Actualmente existen alrededor de 3000. En este libro mencionaremos muchos RFCs.

Para 1989 Internet había crecido tanto que este estilo sumamente informal dejó de ser funcional. Muchos fabricantes ofrecían productos de TCP/IP en ese entonces y no deseaban cambiarlos tan sólo porque 10 investigadores habían concebido una mejor idea. El IAB fue reorganizado de nueva cuenta en el verano de 1989. Los investigadores fueron transferidos a la **IRTF** (**Fuerza de Trabajo para la Investigación sobre Internet**), que fue puesta bajo el mando del IAB, junto con la **IETF** (**Fuerza de Trabajo para la Ingeniería de Internet**). El IAB se renovó con nuevos

miembros, que representaban a un rango más amplio de organizaciones que tan sólo a la comunidad de investigadores. Inicialmente fue un grupo que se autorrenovaba, cuyos miembros servían durante dos años y ellos mismos designaban a sus sucesores. Más tarde se creó la **Sociedad de Internet,** integrada por gente interesada en Internet. En cierto sentido, esta sociedad se asemeja al ACM o al IEEE. Es dirigida por administradores electos que designan a los miembros del IAB.

El propósito de esta división era que la IRTF se concentrara en proyectos de investigación a largo plazo, en tanto que la IETF se encargaba de proyectos de ingeniería a corto plazo. La IETF se dividió en grupos de trabajo, cada uno a cargo de un problema específico. Inicialmente, los líderes de cada grupo se reunían en un comité para dirigir los proyectos de ingeniería. Entre los temas de los grupos de trabajo están nuevas aplicaciones, información de usuario, integración OSI, enrutamiento y direccionamiento, seguridad, administración de redes y estándares. Con el tiempo se formaron tantos grupos de trabajo (más de 70), que se ordenaron por áreas y el líder de cada área formaba parte del comité directivo.

Además, se adoptó un proceso de estandarización más formal, con base en el ISO. Para convertirse en **Estándar Propuesto,** la idea fundamental debía explicarse completamente en un RFC y despertar suficiente interés en la comunidad. Para avanzar a la etapa de **Estándar Borrador,** una implementación funcional debía haber sido rigurosamente probada por al menos dos sitios independientes durante al menos cuatro meses. Si el IAB se convence de que la idea suena lógica y el software funciona, declara que el RFC es un Estándar de Internet. Algunos de estos estándares se han convertido en estándares del DoD (MIL-STD), con lo cual son obligatorios para los proveedores del DoD. En cierta ocasión, David Clark hizo el siguiente comentario, ahora famoso, acerca del proceso de estandarización de Internet: “consenso apretado y código que funcione”.

1.7 UNIDADES MÉTRICAS

Para evitar confusiones, vale la pena indicar de manera explícita que en este libro, como en las ciencias de la computación en general, se utilizan unidades métricas en lugar de las unidades inglesas tradicionales. En la figura 1.39 se muestran los principales prefijos del sistema métrico. Por lo general, los prefijos se abrevian mediante sus primeras letras, con las unidades mayores que uno en mayúsculas (KB, MB, etcétera). Una excepción (por razones históricas) es kbps, de kilobits por segundo. Por lo tanto, una línea de comunicación de 1 Mbps transmite 10^6 bits por segundo y un reloj de 100 pseg (o 100 ps) marca cada 10^{-10} segundos. Dado que tanto mili como micro empiezan con la letra “m”, se tiene que hacer una distinción. Por lo general, “m” es para mili y “μ” (la letra griega mu) es para micro.

También vale la pena señalar que para medir el tamaño de la memoria, de disco, de archivo y de bases de datos, en la práctica común de la industria de la computación las unidades tienen equivalencias ligeramente distintas. En ésta, kilo equivale a 2^{10} (1024) en vez de 10^3 (1000) porque las memorias siempre son potencias de dos. De esta forma, 1 KB de memoria son 1024 bytes, no 1000 bytes. De manera similar, 1 MB de memoria son 2^{20} (1,048,576) bytes, 1 GB de memoria son 2^{30} (1,073,741,824) bytes, y 1 TB de base de datos son 2^{40} (1,099,511,627,776) bytes. No obstante, una línea de comunicación de 1 kbps transmite 1000 bits por segundo y una LAN de

Exp.	Explícito	Prefijo	Exp.	Explícito	Prefijo
10^{-3}	0.001	mili	10^{3}	1,000	Kilo
10^{-6}	0.000001	micro	10^{6}	1,000,000	Mega
10^{-9}	0.000000001	nano	10^{9}	1,000,000,000	Giga
10^{-12}	0.000000000001	pico	10^{12}	1,000,000,000,000	Tera
10^{-15}	0.000000000000001	femto	10^{15}	1,000,000,000,000,000	Peta
10^{-18}	0.000000000000000001	atto	10^{18}	1,000,000,000,000,000,000	Exa
10^{-21}	0.000000000000000000001	zepto	10^{21}	1,000,000,000,000,000,000,000	Zeta
10^{-24}	0.000000000000000000000001	yocto	10^{24}	1,000,000,000,000,000,000,000,000	Yotta

Figura 1-39. Los principales prefijos métricos.

10 Mbps corre a 10,000,000 de bits por segundo debido a que estas velocidades no son potencias de dos. Desgraciadamente, mucha gente mezcla estos dos sistemas, en particular en lo referente a los tamaños de disco. Para evitar la ambigüedad, en este libro utilizaremos los símbolos KB, MB y GB para 2^{10}, 2^{20} y 2^{30} bytes, respectivamente, y los símbolos kbps, Mbps y Gbps para 10^3, 10^6 y 10^9 bits por segundo, respectivamente.

1.8 PANORAMA DEL RESTO DEL LIBRO

Este libro estudia tanto los principios como la práctica de las redes de computadoras. La mayoría de los capítulos inician con un análisis de los principios relevantes, seguido por diversos ejemplos que ilustran estos principios. Por lo general, los ejemplos se toman de Internet y de las redes inalámbricas puesto que ambos son importantes y muy distintos. Donde es necesario, se dan otros ejemplos.

El libro se estructura de acuerdo con el modelo híbrido que se presenta en la figura 1-24. El análisis de la jerarquía de protocolos empieza en el capítulo 2, a partir de la capa más baja. El segundo capítulo proporciona algunos antecedentes en el campo de la comunicación de datos. Se presentan sistemas alámbricos, inalámbricos y satelitales. Este material se relaciona con la capa física, aunque veremos únicamente los aspectos de arquitectura y no los de hardware. También se analizan numerosos ejemplos de la capa física, como la red telefónica pública conmutada, la telefonía celular y la red de televisión por cable.

En el capítulo 3 se presenta la capa de enlace de datos y sus protocolos a través de ejemplos que crecen en complejidad. También se cubre el análisis de estos protocolos. Más tarde se examinan algunos protocolos importantes que se usan con mucha frecuencia, entre ellos HDLC (que se emplea en redes de baja y mediana velocidad) y PPP (que se utiliza en Internet).

El capítulo 4 tiene que ver con la subcapa de acceso al medio, que forma parte de la capa de enlace de datos. El aspecto principal al que se enfrenta esta subcapa es cómo determinar quién uti-

lizará la red cuando ésta consiste en un solo canal compartido, como ocurre en la mayoría de las LANs y en algunas redes satelitales. Se dan muchos ejemplos de LANs alámbricas, LANs inalámbricas (en especial Ethernet), MANs inalámbricas, Bluetooth y redes satelitales. También se analizan los puentes y los conmutadores de enlace de datos.

El capítulo 5 aborda la capa de red, en particular el enrutamiento, con muchos algoritmos de enrutamiento, tanto estáticos como dinámicos. Aun con el uso de buenos algoritmos de enrutamiento, si existe más tráfico del que puede manejar la red, se genera congestión, por lo que analizaremos el tema de la congestión y cómo evitarla. Es aún mejor garantizar una calidad específica en el servicio que tan sólo evitar la congestión. También analizaremos este punto. La conexión de redes heterogéneas para conformar interredes acarrea numerosos problemas que también examinaremos. Daremos una amplia cobertura a la capa de red en Internet.

El capítulo 6 se encarga de la capa de transporte. Gran parte del capítulo se dedica a los protocolos orientados a la conexión, puesto que muchas aplicaciones los necesitan. Se analiza en detalle un ejemplo de servicio de transporte y su implementación. Se proporciona el código para este sencillo ejemplo con el propósito de mostrar cómo se puede implementar. Tanto UDP como TCP, protocolos de transporte de Internet, se abordan en detalle, al igual que sus aspectos de desempeño. Asimismo, veremos aspectos relacionados con las redes inalámbricas.

El capítulo 7 presenta la capa de aplicación, sus protocolos y aplicaciones. El primer tema es el DNS, que es el directorio telefónico de Internet. A continuación trataremos el correo electrónico, junto con un análisis de sus protocolos. Más adelante pasaremos a Web, con explicaciones minuciosas sobre contenido estático, contenido dinámico, lo que sucede tanto en el cliente como en el servidor, protocolos, rendimiento, la Web inalámbrica, entre otros temas. Por último, examinaremos la multimedia en red, con temas como audio de flujo continuo, radio en Internet y vídeo bajo demanda.

El capítulo 8 se relaciona con la seguridad de red. Este tema tiene aspectos que se relacionan con todas las capas, por lo cual es más sencillo abordarlo después de haber explicado minuciosamente todas las capas. El capítulo inicia con una introducción a la criptografía. Más adelante muestra cómo se puede utilizar ésta para garantizar la seguridad en las comunicaciones, el correo electrónico y Web. El libro finaliza con un análisis de algunas áreas en las cuales la seguridad afecta la privacidad, la libertad de expresión, la censura y otros aspectos sociales con los cuales choca directamente.

El capítulo 9 contiene listas de lecturas sugeridas, con comentarios, organizadas por capítulo. Su propósito es ayudar a los lectores que deseen llevar más allá el estudio sobre las redes. El capítulo también tiene una bibliografía alfabética de todas las referencias que se dan en el libro.

El sitio Web del autor puede consultarlo desde:

http://www.pearsonedlatino.com/tanenbaum

el cual contiene una página con vínculos a muchos tutoriales, FAQs, compañías, consorcios industriales, organizaciones profesionales, organizaciones de estándares, tecnologías, documentos y muchas cosas más.

1.9 RESUMEN

Las redes de computadoras se pueden utilizar para diversos servicios, tanto para compañías como para individuos. Para las compañías, las redes de computadoras personales que utilizan servidores compartidos con frecuencia dan acceso a información corporativa. Por lo general, estas redes siguen el modelo cliente-servidor, con estaciones de trabajo clientes en los escritorios de los empleados que acceden a servidores instalados en la sala de máquinas. Para los individuos, las redes ofrecen acceso a una diversidad de recursos de información y entretenimiento. Los individuos acceden a Internet mediante una llamada al ISP a través de un módem, aunque una cantidad creciente de usuarios cuenta con una conexión fija en casa. Un área con gran futuro es la de las redes inalámbricas, con nuevas aplicaciones como acceso móvil al correo electrónico y el comercio móvil.

A grandes rasgos, las redes se pueden dividir en LANs, MANs, WANs e interredes, con sus propias características, tecnologías, velocidades y nichos. Las LANs ocupan edificios y operan a altas velocidades. Las MANs abarcan toda una ciudad, por ejemplo, el sistema de televisión por cable, el cual es utilizado por mucha gente para acceder a Internet. Las WANs se extienden por un país o un continente. Las LANs y las MANs pueden ser o no conmutadas (es decir, no tienen enrutadores); las WANs son conmutadas. Las redes inalámbricas se están volviendo sumamente populares, en especial las LANs inalámbricas. Las redes se interconectan para formar interredes.

El software de red consta de protocolos, que son reglas mediante las cuales se comunican los procesos. Los protocolos son de dos tipos: orientados a la conexión y no orientados a la conexión. La mayoría de las redes soporta jerarquías de protocolos, en la cual cada capa proporciona servicios a las capas superiores a ella y las libera de los detalles de los protocolos que se utilizan en las capas inferiores. Las pilas de protocolos se basan generalmente en el modelo OSI o en el modelo TCP/IP. Ambos modelos tienen capas de red, de transporte y de aplicación, pero difieren en las demás capas. Entre los aspectos de diseño están la multiplexión, el control de flujo y el control de errores. Gran parte del libro está dedicada a los protocolos y su diseño.

Las redes ofrecen servicios a sus usuarios. Los servicios pueden ser orientados a la conexión o no orientados a ésta. En algunas redes se proporciona servicio no orientado a la conexión en una capa y servicio orientado a la conexión en la capa superior.

Las redes bien conocidas incluyen Internet, ATM, Ethernet y la LAN IEEE 802.11 inalámbrica. Internet evolucionó de ARPANET, a la cual se agregaron otras redes para conformar una interred. La Internet actual es en realidad un conjunto de miles de redes, más que de una sola red. El aspecto que la distingue es el uso generalizado de la pila de protocolos TCP/IP. ATM tiene un uso muy extendido en los sistemas telefónicos para el tráfico de datos de larga distancia. Ethernet es la LAN más popular y se utiliza en la mayoría de las compañías y universidades. Por último, las LANs inalámbricas a velocidades sorprendentemente altas (hasta 54 Mbps) comienzan a desplegarse en forma masiva.

Para que varias computadoras se comuniquen entre sí es necesaria una gran cantidad de estandarización, tanto en el software como en el hardware. Organizaciones como la ITU-T, el ISO, el IEEE y el IAB manejan diferentes partes del proceso de estandarización.

PROBLEMAS

1. Imagine que ha entrenado a su San Bernardo, Byron, para que transporte una caja con tres cintas de 8 mm en lugar del barrilito de brandy. (Cuando se llene su disco, usted tendrá una emergencia.) Cada una de estas cintas tiene capacidad de 7 gigabytes. El perro puede trasladarse adondequiera que usted vaya, a una velocidad de 18 km/hora. ¿Para cuál rango de distancias tiene Byron una tasa de datos más alta que una línea de transmisión cuya tasa de datos (sin tomar en cuenta la sobrecarga) es de 150 Mbps?

2. Una alternativa a una LAN es simplemente un enorme sistema de compartición de tiempo con terminales para todos los usuarios. Mencione dos ventajas de un sistema cliente-servidor que utilice una LAN.

3. Dos factores de red ejercen influencia en el rendimiento de un sistema cliente-servidor: el ancho de banda de la red (cuántos bits por segundo puede transportar) y la latencia (cuánto tiempo toma al primer bit llegar del cliente al servidor). Mencione un ejemplo de una red que cuente con ancho de banda y latencia altos. A continuación, mencione un ejemplo de una que cuente con ancho de banda y latencia bajos.

4. ¿Además del ancho de banda y la latencia, qué otros parámetros son necesarios para dar un buen ejemplo de la calidad de servicio ofrecida por una red destinada a tráfico de voz digitalizada?

5. Un factor en el retardo de un sistema de conmutación de paquetes de almacenamiento y reenvío es el tiempo que le toma almacenar y reenviar un paquete a través de un conmutador. Si el tiempo de conmutación es de 10 μseg, ¿esto podría ser un factor determinante en la respuesta de un sistema cliente-servidor en el cual el cliente se encuentre en Nueva York y el servidor en California? Suponga que la velocidad de propagación en cobre y fibra es 2/3 de la velocidad de la luz en el vacío.

6. Un sistema cliente-servidor utiliza una red satelital, con el satélite a una altura de 40,000 km. ¿Cuál es el retardo en respuesta a una solicitud, en el mejor de los casos?

7. En el futuro, cuando cada persona tenga una terminal en casa conectada a una red de computadoras, serán posibles las consultas públicas instantáneas sobre asuntos legislativos pendientes. Con el tiempo, las legislaturas existentes podrían eliminarse, para dejar que la voluntad popular se exprese directamente. Los aspectos positivos de una democracia directa como ésta son bastante obvios; analice algunos de los aspectos negativos.

8. Cinco enrutadores se van a conectar en una subred de punto a punto. Los diseñadores podrían poner una línea de alta velocidad, de mediana velocidad, de baja velocidad o ninguna línea, entre cada par de enrutadores. Si toma 100 ms de tiempo de la computadora generar e inspeccionar cada topología, ¿cuánto tiempo tomará inspeccionarlas todas?

9. Un grupo de $2^n - 1$ enrutadores están interconectados en un árbol binario centralizado, con un enrutador en cada nodo del árbol. El enrutador i se comunica con el enrutador j enviando un mensaje a la raíz del árbol. A continuación, la raíz manda el mensaje al enrutador j. Obtenga una expresión aproximada de la cantidad media de saltos por mensaje para un valor grande de n, suponiendo que todos los pares de enrutadores son igualmente probables.

10. Una desventaja de una subred de difusión es la capacidad que se desperdicia cuando múltiples *hosts* intentan acceder el canal al mismo tiempo. Suponga, por ejemplo, que el tiempo se divide en ranuras discretas, y que cada uno de los *hosts* n intenta utilizar el canal con probabilidad p durante cada parte. ¿Qué fracción de las partes se desperdicia debido a colisiones?

11. Mencione dos razones para utilizar protocolos en capas.

12. Al presidente de Specialty Paint Corp. se le ocurre la idea de trabajar con una compañía cervecera local para producir una lata de cerveza invisible (como medida para reducir los desechos). El presidente indica a su departamento legal que analice la situación, y éste a su vez pide ayuda al departamento de ingeniería. De esta forma, el ingeniero en jefe se reúne con su contraparte de la otra compañía para discutir los aspectos técnicos del proyecto. A continuación, los ingenieros informan los resultados a sus respectivos departamentos legales, los cuales a su vez se comunican vía telefónica para ponerse de acuerdo en los aspectos legales. Por último, los dos presidentes corporativos se ponen de acuerdo en la parte financiera del proyecto. ¿Éste es un ejemplo de protocolo con múltiples capas semejante al modelo OSI?

13. ¿Cuál es la diferencia principal entre comunicación orientada a la conexión y no orientada a ésta?

14. Dos redes proporcionan servicio confiable orientado a la conexión. Una de ellas ofrece un flujo confiable de bytes y la otra un flujo confiable de mensajes. ¿Son idénticas? Si es así, ¿por qué se hace la distinción? Si no son idénticas, mencione un ejemplo de algo en que difieran.

15. ¿Qué significa "negociación" en el contexto de protocolos de red? Dé un ejemplo.

16. En la figura 1-19 se muestra un servicio. ¿Hay algún otro servicio implícito en la figura? Si es así, ¿dónde? Si no lo hay, ¿por qué no?

17. En algunas redes, la capa de enlace de datos maneja los errores de transmisión solicitando que se retransmitan las tramas dañadas. Si la probabilidad de que una trama se dañe es p, ¿cuál es la cantidad media de transmisiones requeridas para enviar una trama? Suponga que las confirmaciones de recepción nunca se pierden.

18. ¿Cuál de las capas OSI maneja cada uno de los siguientes aspectos?:
(a) Dividir en tramas el flujo de bits transmitidos.
(b) Determinar la ruta que se utilizará a través de la subred.

19. Si la unidad que se transmite al nivel de enlace de datos se denomina trama y la que se transmite al nivel de red se llama paquete, ¿las tramas encapsulan paquetes o los paquetes encapsulan tramas? Explique su respuesta?

20. Un sistema tiene una jerarquía de protocolos de n capas. Las aplicaciones generan mensajes con una longitud de M bytes. En cada una de las capas se agrega un encabezado de h bytes. ¿Qué fracción del ancho de banda de la red se llena con encabezados?

21. Mencione dos similitudes entre los modelos de referencia OSI y TCP/IP. A continuación mencione dos diferencias entre ellos.

22. ¿Cuál es la principal diferencia entre TCP y UDP?

23. La subred de la figura 1-25(b) se diseñó para resistir una guerra nuclear. ¿Cuántas bombas serían necesarias para partir los nodos en dos conjuntos inconexos? Suponga que cualquier bomba destruye un nodo y todos los enlaces que se conectan a él.

24. Internet está duplicando su tamaño aproximadamente cada 18 meses. Aunque no se sabe a ciencia cierta, una estimación indica que en el 2001 había 100 millones de *hosts* en Internet. Utilice estos datos para calcular la cantidad esperada de *hosts* para el año 2010. ¿Cree que esto es real? Explique por qué.

25. Cuando un archivo se transfiere entre dos computadoras, pueden seguirse dos estrategias de confirmación de recepción. En la primera, el archivo se divide en paquetes, y el receptor confirma la recepción de cada uno de manera individual, aunque no confirma la recepción del archivo como un todo. En contraste, en la segunda estrategia la recepción de los paquetes no se confirma de manera individual, sino la del archivo completo. Comente las dos estrategias.

26. ¿Por qué ATM utiliza celdas pequeñas de longitud fija?

27. ¿Qué tan grande era un bit, en metros, en el estándar 802.3 original? Utilice una velocidad de transmisión de 10 Mbps y suponga que la velocidad de propagación en cable coaxial es 2/3 la velocidad de la luz en el vacío.

28. Una imagen tiene 1024×768 píxeles con 3 bytes/píxel. Suponga que la imagen no se encuentra comprimida. ¿Cuánto tiempo tomará transmitirla sobre un canal de módem de 56 kpbs? ¿Sobre un módem de cable de 1 Mbps? ¿Sobre una red Ethernet a 10 Mbps? ¿Sobre una red Ethernet a 100 Mbps?

29. Ethernet y las redes inalámbricas tienen algunas similitudes y diferencias. Una propiedad de Ethernet es que sólo se puede transmitir una trama a la vez sobre una red de este tipo. ¿El 802.11 comparte esta propiedad con Ethernet? Comente su respuesta.

30. Las redes inalámbricas son fáciles de instalar, y ello las hace muy económicas puesto que los costos de instalación eclipsan por mucho los costos del equipo. No obstante, también tienen algunas desventajas. Mencione dos de ellas.

31. Cite dos ventajas y dos desventajas de contar con estándares internacionales para los protocolos de red.

32. Cuando un sistema tiene una parte fija y una parte removible (como ocurre con una unidad de CD-ROM y el CD-ROM), es importante que exista estandarización en el sistema, con el propósito de que las diferentes compañías puedan fabricar tanto la parte removible como la fija y todo funcione en conjunto. Mencione tres ejemplos ajenos a la industria de la computación en donde existan estándares internacionales. Ahora mencione tres áreas donde no existan.

33. Haga una lista de sus actividades cotidianas en las cuales intervengan las redes de computadoras. ¿De qué manera se alteraría su vida si estas redes fueran súbitamente desconectadas?

34. Averigüe cuáles redes se utilizan en su escuela o lugar de trabajo. Describa los tipos de red, las topologías y los métodos de conmutación que utilizan.

35. El programa *ping* le permite enviar un paquete de prueba a un lugar determinado y medir cuánto tarda en ir y regresar. Utilice *ping* para ver cuánto tiempo toma llegar del lugar donde se encuentra hasta diversos lugares conocidos. Con los resultados, trace el tiempo de tránsito sobre Internet como una función de la distancia. Lo más adecuado es utilizar universidades, puesto que la ubicación de sus servidores se conoce con mucha precisión. Por ejemplo, *berkeley.edu* se encuentra en Berkeley, California; *mit.edu* se localiza en Cambridge, Massachusetts; *vu.nl* está en Amsterdam, Holanda; *www.usyd.edu.au* se encuentra en Sydney, Australia, y *www.uct.ac.za* se localiza en Cape Town, Sudáfrica.

36. Vaya al sitio Web de la IETF, *www.ietf.org,* y entérese de lo que hacen ahí. Elija un proyecto y escriba un informe de media página acerca del problema y la solución que propone.

37. La estandarización es sumamente importante en el mundo de las redes. La ITU y la ISO son las principales organizaciones oficiales encargadas de la estandarización. Vaya a los sitios Web de estas organiza-

ciones, en *www.itu.org* y *www.iso.org,* respectivamente, y analice el trabajo de estandarización que realizan. Escriba un breve informe sobre las cosas que han estandarizado.

38. Internet está conformada por una gran cantidad de redes. Su disposición determina la topología de Internet. En línea se encuentra una cantidad considerable de información acerca de la topología de Internet. Utilice un motor de búsqueda para investigar más sobre la topología de Internet y escriba un breve informe sobre sus resultados.

2

LA CAPA FÍSICA

En este capítulo analizaremos la capa que está en la parte más baja de la jerarquía de la figura 1-24. Dicha capa define las interfaces mecánica, eléctrica y de temporización de la red. Comenzaremos con un análisis teórico de la transmisión de datos, el cual nos llevará a descubrir que la Madre Naturaleza establece límites en lo que se puede enviar a través de un canal.

Después trataremos tres tipos de medios de transmisión: dirigidos (cable de cobre y fibra óptica), inalámbricos (radio terrestre) y por satélite. Este material proporcionará información a fondo de las principales tecnologías de transmisión que se utilizan en las redes actuales.

El resto del capítulo se dedicará a dar tres ejemplos de sistemas de comunicación que se utilizan en la práctica en las redes de computadora de área amplia: el sistema telefónico (fijo), el sistema de telefonía móvil y el sistema de televisión por cable. Los tres utilizan una red dorsal de fibra óptica, pero están organizados de diferente manera y utilizan tecnologías distintas en la última milla (la conexión hacia el cliente).

2.1 LA BASE TEÓRICA DE LA COMUNICACIÓN DE DATOS

Mediante la variación de algunas propiedades físicas, como el voltaje o la corriente, es posible transmitir información a través de cables. Al representar el valor de este voltaje o corriente como una función simple del tiempo, $f(t)$, podemos modelar el comportamiento de la señal y analizarlo matemáticamente. Este análisis es el tema de las siguientes secciones.

2.1.1 El análisis de Fourier

A principios del siglo XIX, el matemático francés Jean-Baptiste Fourier probó que cualquier función periódica de comportamiento razonable, $g(t)$ con un periodo T, se puede construir sumando una cantidad (posiblemente infinita) de senos y cosenos:

$$g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n \operatorname{sen}(2\pi nft) + \sum_{n=1}^{\infty} b_n \cos(2\pi nft) \qquad (2\text{-}1)$$

donde $f = 1/T$ es la frecuencia fundamental, a_n y b_n son las amplitudes de seno y coseno de los n-ésimos (términos) **armónicos** y c es una constante. Tal descomposición se conoce como **serie de Fourier.** A partir de ella, es posible reconstruir la función, es decir, si se conoce el periodo T y se dan las amplitudes, la función original del tiempo puede encontrarse realizando las sumas que se muestran en la ecuación (2-1).

Una señal de datos que tenga una duración finita (la cual todas poseen) se puede manejar con sólo imaginar que el patrón se repite una y otra vez por siempre (es decir, el intervalo de T a $2T$ es el mismo que de 0 a T, etcétera).

Las amplitudes a_n se pueden calcular para cualquier $g(t)$ dada multiplicando ambos lados de la ecuación (2-1) por sen($2\pi kft$) y después integrando de 0 a T. Puesto que

$$\int_0^T \operatorname{sen}(2\pi kft)\operatorname{sen}(2\pi nft)\,dt = \begin{cases} 0 \text{ para } k \neq n \\ T/2 \text{ para } k = n \end{cases}$$

sólo un término de la sumatoria perdura: a_n. La sumatoria de b_n desaparece por completo. De manera similar, al multiplicar la ecuación (2-1) por cos($2\pi kft$) e integrando entre 0 y T, podemos derivar b_n. Con sólo integrar ambos lados de la ecuación como está, podemos encontrar c. Los resultados de realizar estas operaciones son los siguientes:

$$a_n = \frac{2}{T}\int_0^T g(t)\operatorname{sen}(2\pi nft)\,dt \qquad b_n = \frac{2}{T}\int_0^T g(t)\cos(2\pi nft)\,dt \qquad c = \frac{2}{T}\int_0^T g(t)\,dt$$

2.1.2 Señales de ancho de banda limitado

Para ver cómo se relaciona todo esto con la comunicación de datos, consideremos un ejemplo específico: la transmisión del carácter “b” ASCII codificado en un byte de 8 bits. El patrón de bits que se va a transmitir es 01100010. La parte izquierda de la figura 2-1(a) muestra la salida de voltaje que produce la computadora transmisora. El análisis de Fourier de la señal produce los coeficientes:

$$a_n = \frac{1}{\pi n}\left[\cos(\pi n/4) - \cos(3\pi n/4) + \cos(6\pi n/4) - \cos(7\pi n/4)\right]$$

$$b_n = \frac{1}{\pi n}\left[\operatorname{sen}(3\pi n/4) - \operatorname{sen}(\pi n/4) + \operatorname{sen}(7\pi n/4) - \operatorname{sen}(6\pi n/4)\right]$$

$$c = 3/4$$

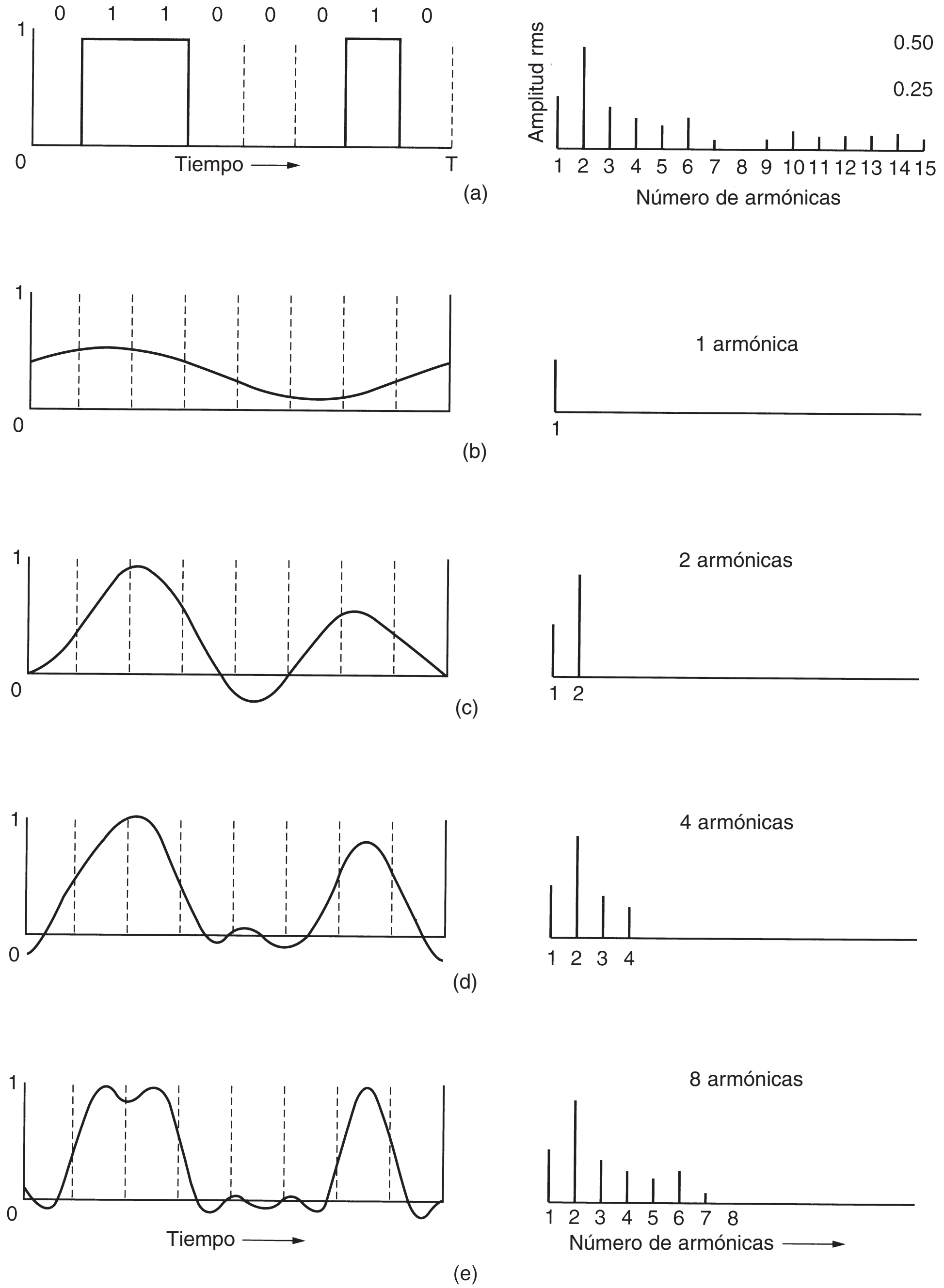


Figura 2-1. (a) Una señal binaria y sus amplitudes de raíz cuadrada media de Fourier. (b)-(e) Aproximaciones sucesivas a la señal original.

En el lado derecho de la figura 2-1(a) se muestran las amplitudes de raíz cuadrada media, $\sqrt{a_n^2 + b_n^2}$, para los primeros términos. Estos valores son importantes porque sus cuadrados son proporcionales a la energía transmitida en la frecuencia correspondiente.

Ninguna instalación transmisora puede transmitir señales sin perder cierta potencia en el proceso. Si todos los componentes de Fourier disminuyeran en la misma proporción, la señal resultante se reduciría en amplitud, pero no se distorsionaría [es decir, tendría la misma forma cuadrada que tiene en la figura 2-1(a)]. Desgraciadamente, todas las instalaciones de transmisión disminuyen los distintos componentes de Fourier en diferente grado, lo que provoca distorsión. Por lo general, las amplitudes se transmiten sin ninguna disminución desde 0 hasta cierta frecuencia f_c [medida en ciclos/seg o Hertz (Hz)], y todas las frecuencias que se encuentren por arriba de esta frecuencia de corte serán atenuadas. El rango de frecuencias que se transmiten sin atenuarse con fuerza se conoce como **ancho de banda.** En la práctica, el corte en realidad no es abrupto, por lo que con frecuencia el ancho de banda ofrecido va desde 0 hasta la frecuencia en la que el valor de la amplitud es atenuado a la mitad de su valor original.

El ancho de banda es una propiedad física del medio de transmisión y por lo general depende de la construcción, grosor y longitud de dicho medio. En algunos casos, se introduce un filtro en el circuito para limitar la cantidad de ancho de banda disponible para cada cliente. Por ejemplo, un cable de teléfono podría tener un ancho de banda de 1 MHz para distancias cortas, pero las compañías telefónicas agregan un filtro que restringe a cada cliente a aproximadamente 3100 Hz. Este ancho de banda es adecuado para el lenguaje inteligible y mejora la eficiencia del sistema al limitar a los usuarios en el uso de los recursos.

Ahora consideremos cómo luciría la señal de la figura 2-1(a) si el ancho de banda fuera tan lento que sólo las frecuencias más bajas se transmitieran [es decir, si la función fuera aproximada por los primeros términos de la ecuación 2-1(a)]. La figura 2-1(b) muestra la señal que resulta de un canal que permite que sólo pase la primera armónica (la fundamental, f). De manera similar, la figura 2-1(c)-(e) muestra el espectro y las funciones reconstruidas de canales de ancho de banda más grande.

Dada una tasa de bits de b bits/seg, el tiempo requerido para enviar 8 bits (por ejemplo) 1 bit a la vez es $8/b$ seg, por lo que la frecuencia de la primera armónica es $b/8$ Hz. Una línea telefónica normal, llamada con frecuencia **línea con calidad de voz,** tiene una frecuencia de corte introducida de manera artificial arriba de 3000 Hz. Esta restricción significa que el número de armónicas más altas que pasan es de aproximadamente $3000/(b/8)$ o $24{,}000/b$ (el corte no es abrupto).

Para algunas tasas de datos, los números resultan como se muestra en la figura 2-2. A partir de estos números, queda claro que tratar de transmitir a 9600 bps por una línea telefónica transformará la figura 2-1(a) en algo similar a lo que se muestra en la figura 2-1(c), lo que dificulta la recepción precisa del flujo de bits binarios original. Debería ser obvio que a tasas de datos mucho mayores que 38.4 kbps, no hay la menor esperanza para las señales *binarias*, aun si la transmisión se encuentra completamente libre de ruidos. En otras palabras, limitar el ancho de banda limita la tasa de datos, incluso en canales perfectos. Sin embargo, existen esquemas de codificación refinados que utilizan diferentes niveles de voltaje y pueden alcanzar tasas de datos mayores. Este tema lo trataremos con mayor detalle más adelante en el capítulo.

Bps	T (mseg)	Primera armónica (Hz)	# de armónicas enviadas
300	26.67	37.5	80
600	13.33	75	40
1200	6.67	150	20
2400	3.33	300	10
4800	1.67	600	5
9600	0.83	1200	2
19200	0.42	2400	1
38400	0.21	4800	0

Figura 2-2. Relación entre tasa de datos y armónicas.

2.1.3 La tasa de datos máxima de un canal

En 1924, un ingeniero de AT&T, Henry Nyquist, se dio cuenta de que incluso un canal perfecto tiene una capacidad de transmisión finita. Derivó una ecuación que expresa la tasa de datos máxima para un canal sin ruido de ancho de banda finito. En 1948, Claude Shannon continuó el trabajo de Nyquist y lo extendió al caso de un canal sujeto a ruido aleatorio (es decir, termodinámico) (Shannon, 1948). Sólo resumiremos brevemente sus ahora clásicos resultados.

Nyquist probó que si se pasa una señal cualquiera a través de un filtro pasa-bajas de ancho de banda H, la señal filtrada se puede reconstruir por completo tomando sólo $2H$ muestras (exactas) por segundo. No tiene sentido muestrear la línea a una rapidez mayor que $2H$ veces por segundo porque los componentes de mayor frecuencia que tal muestreo puede recuperar ya se han filtrado. Si la señal consiste en V niveles discretos, el teorema de Nyquist establece:

$$\text{tasa de datos máxima} = 2H \log_2 V \text{ bits/seg}$$

Por ejemplo, un canal sin ruido de 3 kHz no puede transmitir señales binarias (es decir, de dos niveles) a una tasa mayor que 6000 bps.

Hasta aquí sólo hemos considerado canales sin ruido. Si el ruido aleatorio está presente, la situación se deteriora rápidamente. Y el ruido aleatorio (térmico) siempre está presente debido al movimiento de las moléculas del sistema. La cantidad de ruido térmico presente se mide por la relación entre la potencia de la señal y la potencia del ruido, llamada **relación señal a ruido.** Si indicamos la potencia de la señal con una S y la potencia del ruido con N, la relación señal a ruido es S/N. Por lo general, la relación misma no se expresa; en su lugar, se da la cantidad $10 \log_{10} S/N$. Estas unidades se conocen como **decibeles** (dB). Una relación S/N de 10 es 10 dB, una relación de 100 es 20 dB, una de 1000 es 30 dB, y así sucesivamente. Los fabricantes de amplificadores estereofónicos a menudo caracterizan el ancho de banda (rango de frecuencia) en el cual su producto es lineal dando la frecuencia de 3 dB en cada extremo. Éstos son los puntos a los que el factor de amplificación ha sido dividido (puesto que $\log_{10} 3 \approx 0.5$).

El resultado principal de Shannon es que la tasa de datos máxima de un canal ruidoso cuyo ancho de banda es *H* Hz y cuya relación señal a ruido es *S/N*, está dada por

$$\text{número máximo de bits/seg} = H \log_2 (1 + S/N)$$

Por ejemplo, un canal con un ancho de banda de 3000 Hz y con una relación señal a ruido térmico de 30 dB (los parámetros típicos de la parte analógica del sistema telefónico) no puede transmitir más allá de 30,000 bps, sin importar cuántos niveles de señal se utilicen, ni con qué frecuencia se tomen los muestreos. El resultado de Shannon se dedujo aplicando argumentos de la teoría de la información y es válido para cualquier canal sujeto a ruido térmico. Los ejemplos contrarios se deben clasificar en la misma categoría de las máquinas de movimiento perpetuo. Sin embargo, cabe señalar que éste solamente es un límite superior y que los sistemas reales rara vez lo alcanzan.

2.2 MEDIOS DE TRANSMISIÓN GUIADOS

El propósito de la capa física es transportar un flujo de datos puro de una máquina a otra. Es posible utilizar varios medios físicos para la transmisión real. Cada uno tiene su propio nicho en términos de ancho de banda, retardo, costo y facilidad de instalación y mantenimiento. Los medios se clasifican de manera general en medios guiados, como cable de cobre y fibra óptica, y medios no guiados, como radio y láser a través del aire. Analizaremos estos temas en las siguientes secciones.

2.2.1 Medios magnéticos

Una de las formas más comunes para transportar datos de una computadora a otra es almacenarlos en cintas magnéticas o medios extraíbles (por ejemplo, DVDs grabables), transportar físicamente la cinta o los discos a la máquina de destino y leer dichos datos ahí. Si bien este método no es tan avanzado como utilizar un satélite de comunicaciones geosíncrono, con frecuencia es más rentable, especialmente para aplicaciones en las que un ancho de banda alto o el costo por bit transportado es un factor clave.

Un cálculo simple aclarará este punto. Una cinta Ultrium estándar puede almacenar 200 gigabits. Una caja de $60 \times 60 \times 60$ cm puede contener aproximadamente 1000 de estas cintas, con una capacidad total de 200 terabytes, o 1600 terabits (1.6 petabits). Una caja de cintas puede enviarse a cualquier parte de Estados Unidos en 24 horas por Federal Express y otras compañías. El ancho de banda efectivo de esta transmisión es de 1600 terabits/86,400 seg o 19 Gbps. Si el destino está a sólo una hora por carretera, el ancho de banda se incrementa a casi 400 Gbps. Ninguna red de computadoras puede aprovechar esto.

En el caso de un banco que diariamente tiene que respaldar muchos gigabytes de datos en una segunda máquina (para poder continuar en caso de que suceda alguna inundación o un terremoto), es probable que ninguna otra tecnología de transmisión pueda siquiera acercarse en rendimiento a la cinta magnética. Es cierto que la rapidez de las redes se está incrementando, pero también las densidades de las cintas.

Si vemos ahora el costo, obtendremos un panorama similar. El costo de una cinta Ultrium es de aproximadamente $40 cuando se compra al mayoreo. Una cinta puede reutilizarse al menos 10 veces, por lo que el costo de la cinta podría ser de $4000 por caja, por uso. Agreguemos otros $1000 por el envío (probablemente menos), y tenemos un costo de más o menos $5000 por almacenar 200 TB. Esto equivale a 3 centavos por cada gigabyte. Ninguna red puede superar esto. La moraleja es:

Nunca subestime el ancho de banda de una camioneta repleta de cintas que va a toda velocidad por la carretera

2.2.2 Par trenzado

Aunque las características del ancho de banda de una cinta magnética son excelentes, las de retardo son pobres. El tiempo de transmisión se mide en minutos u horas, no en milisegundos. Para muchas aplicaciones se necesita una conexión en línea. Uno de los medios de transmisión más viejos, y todavía el más común, es el **cable de par trenzado.** Éste consiste en dos alambres de cobre aislados, por lo regular de 1 mm de grueso. Los alambres se trenzan en forma helicoidal, igual que una molécula de DNA. Esto se hace porque dos alambres paralelos constituyen una antena simple. Cuando se trenzan los alambres, las ondas de diferentes vueltas se cancelan, por lo que la radiación del cable es menos efectiva.

La aplicación más común del cable de par trenzado es en el sistema telefónico. Casi todos los teléfonos están conectados a la compañía telefónica mediante un cable de par trenzado. La distancia que se puede recorrer con estos cables es de varios kilómetros sin necesidad de amplificar las señales, pero para distancias mayores se requieren repetidores. Cuando muchos cables de par trenzado recorren de manera paralela distancias considerables, como podría ser el caso de los cables de un edificio de departamentos que van hacia la compañía telefónica, se suelen atar en haces y se cubren con una envoltura protectora. Los cables dentro de estos haces podrían sufrir interferencias si no estuvieran trenzados. En algunos lugares del mundo en donde las líneas telefónicas se instalan en la parte alta de los postes, se observan frecuentemente dichos haces, de varios centímetros de diámetro.

Los cables de par trenzado se pueden utilizar para transmisión tanto analógica como digital. El ancho de banda depende del grosor del cable y de la distancia que recorre; en muchos casos pueden obtenerse transmisiones de varios megabits/seg, en distancias de pocos kilómetros. Debido a su comportamiento adecuado y bajo costo, los cables de par trenzado se utilizan ampliamente y es probable que permanezcan por muchos años.

Hay varios tipos de cableado de par trenzado, dos de los cuales son importantes para las redes de computadoras. Los cables de par trenzado **categoría 3** consisten en 2 alambres aislados que se trenzan de manera delicada. Cuatro de estos pares se agrupan por lo regular en una envoltura de plástico para su protección. Antes de 1988, la mayoría de los edificios de oficinas tenía un cable de categoría 3 que iba desde un **gabinete de cableado** central en cada piso hasta cada oficina. Este esquema permitió que hasta cuatro teléfonos comunes o dos teléfonos de múltiples líneas en cada oficina se conectaran con el equipo de la compañía telefónica en el gabinete de cableado.

A comienzos de 1988 se introdujeron los cables de par trenzado **categoría 5** más avanzados. Son similares a los de la categoría 3, pero con más vueltas por centímetro, lo que produce una menor diafonía y una señal de mejor calidad a distancias más largas. Esto los hace más adecuados para una comunicación más rápida entre computadoras. Las siguientes son las categorías 6 y 7, que tienen capacidad para manejar señales con anchos de banda de 250 y 600 MHz, respectivamente (en comparación con los 16 y 100 MHz de las categorías 3 y 5, respectivamente).

Todos estos tipos de cableado comúnmente se conocen como **UTP** (**Par Trenzado sin Blindaje**), en comparación con los cables de par trenzado costosos, blindados y voluminosos que IBM introdujo a principios de la década de 1980, los cuales no ganaron popularidad fuera de las instalaciones de IBM. En la figura 2-3 se muestra un cableado de par trenzado.



Figura 2-3. (a) UTP categoría 3. (b) UTP categoría 5.

2.2.3 Cable coaxial

Otro medio común de transmisión es el **cable coaxial** (conocido frecuentemente tan sólo como “coax”). Este cable tiene mejor blindaje que el de par trenzado, así que puede abarcar tramos más largos a velocidades mayores. Hay dos clases de cable coaxial que son las más utilizadas. Una clase: el cable de 50 ohms, se usa por lo general para transmisión digital. La otra clase, el cable de 75 ohms, se utiliza comúnmente para la transmisión analógica y la televisión por cable, pero se está haciendo cada vez más importante con el advenimiento de Internet a través de cable. Esta distinción se basa en hechos históricos, más que en técnicos (por ejemplo, las antenas antiguas de dipolos tenían una impedancia de 300 ohms y era fácil utilizar los transformadores adaptadores de impedancia 4:1).

Un cable coaxial consiste en un alambre de cobre rígido como núcleo, rodeado por un material aislante. El aislante está forrado con un conductor cilíndrico, que con frecuencia es una malla de tejido fuertemente trenzado. El conductor externo se cubre con una envoltura protectora de plástico. En la figura 2-4 se muestra una vista en corte por capas de un cable coaxial.

La construcción y el blindaje del cable coaxial le confieren una buena combinación de ancho de banda alto y excelente inmunidad al ruido. El ancho de banda posible depende de la calidad y longitud del cable, y de la relación señal a ruido de la señal de datos. Los cables modernos tienen un ancho de banda de cerca de 1 GHz. Los cables coaxiales solían ser ampliamente usados en el sistema telefónico para las líneas de larga distancia, pero en la actualidad han sido reemplazados por la fibra óptica en rutas de distancias considerables. Sin embargo, el cable coaxial aún se utiliza ampliamente en la televisión por cable y en las redes de área metropolitana.

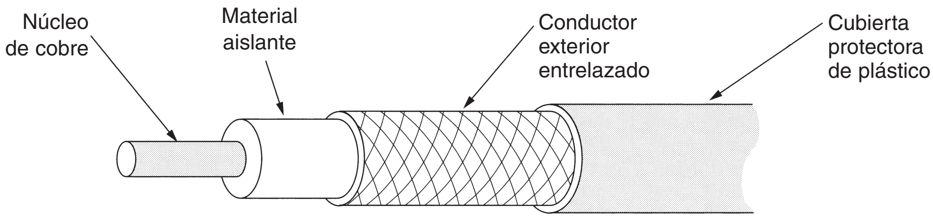


Figura 2-4. Un cable coaxial.

2.2.4 Fibra óptica

Muchas personas de la industria de la computación se enorgullecen de lo rápido que está mejorando la tecnología en esta área. La PC original de IBM (1981) se ejecutaba a una velocidad de reloj de 4.77 MHz. Veinte años más tarde, las PCs pueden correr a 2 GHz, con un factor de ganancia de 20 por década. No está nada mal.

En el mismo periodo, la comunicación de datos de área amplia pasó de 56 kbps (ARPANET) a 1 Gbps (comunicación óptica moderna), con un factor de ganancia de más de 125 por década, y al mismo tiempo la tasa de error pasó de 10^{-5} por bit hasta casi cero.

Además, las CPUs individuales están empezando a aproximarse a límites físicos, como la velocidad de la luz y los problemas de la disipación de calor. En contraste, con la tecnología *actual* de fibras, el ancho de banda alcanzable ciertamente está por encima de los 50,000 Gbps (50 Tbps) y muchas personas se están esforzando arduamente para encontrar mejores tecnologías y materiales. El límite práctico de señalización actual de aproximadamente 10 Gbps se debe a nuestra incapacidad para convertir con mayor rapidez las señales eléctricas a ópticas, aunque en el laboratorio se han alcanzado hasta 100 Gbps en una sola fibra.

En la competencia entre la computación y la comunicación, esta última ganó. La generación de científicos e ingenieros de computación acostumbrados a pensar en términos de los bajos límites de Nyquist y Shannon impuestos por el alambre de cobre aún no ha comprendido todas las implicaciones del ancho de banda prácticamente infinito (aunque no sin un costo). El nuevo sentido común debería ser que todas las computadoras son desesperadamente lentas y que las redes deberían tratar de evitar las tareas de cómputo a cualquier precio, sin importar cuánto ancho de banda se desperdicie. En esta sección analizaremos la fibra óptica para ver cómo funciona esa tecnología de transmisión.

Un sistema de transmisión óptico tiene tres componentes: la fuente de luz, el medio de transmisión y el detector. Convencionalmente, un pulso de luz indica un bit 1 y la ausencia de luz indica un bit 0. El medio de transmisión es una fibra de vidrio ultradelgada. El detector genera un pulso eléctrico cuando la luz incide en él. Al agregar una fuente de luz en un extremo de una fibra óptica y un detector en el otro, se tiene un sistema de transmisión de datos unidireccional que acepta una señal eléctrica, la convierte y transmite mediante pulsos de luz y, luego, reconvierte la salida a una señal eléctrica en el extremo receptor.

Este sistema de transmisión tendría fugas de luz y sería inútil en la práctica excepto por un principio interesante de la física. Cuando un rayo de luz pasa por un medio a otro —por ejemplo, de sílice fundida al aire—, el rayo se refracta (se dobla) en la frontera de la sílice y el aire, como se muestra en la figura 2-5(a). En ella vemos un rayo de luz que incide en la frontera con un ángulo α_1 y que emerge con un ángulo β_1. El grado de refracción depende de las propiedades de los dos medios (en particular sus índices de refracción). Para ángulos con incidencias mayores de ciertos valores críticos, la luz se refracta nuevamente a la sílice; ninguna parte de él escapa al aire. Por lo tanto, un rayo de luz que incide en un ángulo mayor o igual que el crítico queda atrapado dentro de la fibra, como se muestra en la figura 2-5(b), y se puede propagar por varios kilómetros prácticamente sin pérdida.

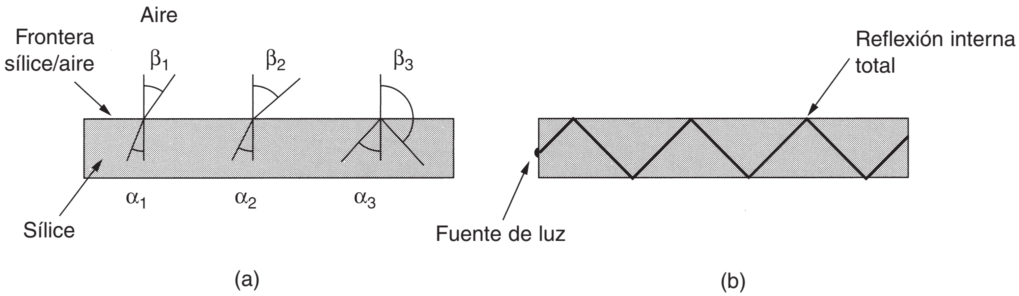


Figura 2-5. (a) Tres ejemplos de un rayo de luz procedente del interior de una fibra de sílice que incide sobre la frontera de la sílice y el aire con diferentes ángulos. (b) Luz atrapada por reflexión interna total.

El diagrama de la segunda figura únicamente muestra un rayo atrapado, pero puesto que cualquier rayo de luz que incida en la frontera con un ángulo mayor que el crítico se reflejará internamente, muchos rayos estarán rebotando con ángulos diferentes. Se dice que cada rayo tiene un **modo** diferente, por lo que una fibra que tiene esta propiedad se denomina **fibra multimodo.**

Por otro lado, si el diámetro de la fibra se reduce a unas cuantas longitudes de onda de luz, la fibra actúa como una guía de ondas y la luz se puede propagar sólo en línea recta, sin rebotar, lo cual da como resultado una **fibra monomodo.** Las fibras monomodo son más caras, pero se pueden utilizar en distancias más grandes. Las fibras monomodo disponibles en la actualidad pueden transmitir datos a 50 Gbps a una distancia de 100 km sin amplificación. En el laboratorio se han logrado tasas de datos todavía mayores a distancias más cortas.

Transmisión de la luz a través de fibra óptica

Las fibras ópticas se hacen de vidrio, que a su vez se fabrica con arena, una materia debajo costo disponible en cantidades ilimitadas. La fabricación de vidrio era conocida por los antiguos egipcios, pero su vidrio no tenía más de 1 mm de espesor, porque de lo contrario la luz no podía atravesarlo. Durante el Renacimiento se forjó un vidrio suficientemente transparente para utilizarlo en ventanas. El vidrio utilizado para fabricar fibras ópticas modernas es tan transparente que si

el océano estuviera lleno de éste en lugar de agua, el fondo del mar sería tan visible desde la superficie como lo es el suelo desde un avión en un día claro.

La atenuación de la luz dentro del vidrio depende de la longitud de onda de la luz (así como de algunas propiedades físicas del vidrio). En la figura 2-6 se muestra la atenuación para la clase de vidrio que se usa en las fibras, en decibeles por kilómetro lineal de fibra. La atenuación en decibeles está dada por la fórmula:

$$\text{Atenuación en decibeles} = 10 \log_{10} \frac{\text{potencia transmitida}}{\text{potencia recibida}}$$

Por ejemplo, un factor de pérdida de dos da como resultado una atenuación de $10 \log_{10} 2 = 3$ dB. La figura muestra la parte cercana al infrarrojo del espectro, que es la que se utiliza en la práctica. La luz visible tiene longitudes de onda ligeramente más cortas, de 0.4 a 0.7 micras (1 micra es 10^{-6} metros). Los puristas de la métrica se referirían a estas longitudes de onda como 400 nm a 700 nm, pero nosotros nos apegaremos al uso tradicional.

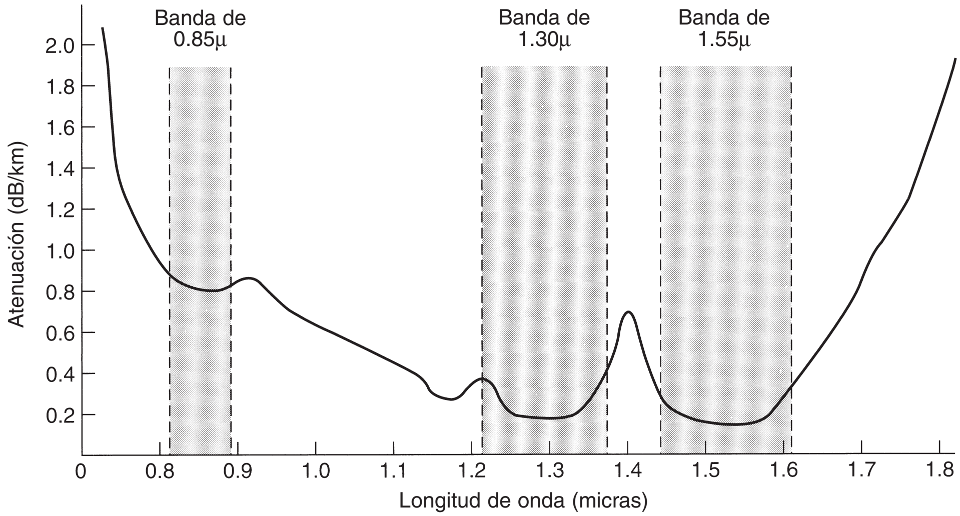


Figura 2-6. Atenuación de la luz dentro de una fibra en la región de infrarrojo.

Para las comunicaciones se utilizan tres bandas de longitud de onda, las cuales se centran en 0.85, 1.30 y 1.55 micras, respectivamente. Las últimas dos tienen buenas propiedades de atenuación (una pérdida de menos de 5% por kilómetro). La banda de 0.85 micras tiene una atenuación más alta, pero a esa longitud de onda, los láseres y los componentes electrónicos se pueden fabricar con el mismo material (arseniuro de galio). Las tres bandas tienen una anchura de entre 25,000 y 30,000 GHz.

La longitud de los pulsos de luz transmitidos por una fibra aumenta conforme se propagan. Este fenómeno se llama **dispersión cromática.** La magnitud de ésta depende de la longitud de

onda. Una forma de evitar que se encimen estos pulsos dispersos es incrementar la distancia entre ellos, pero esto solamente se puede hacer reduciendo la tasa de transmisión. Por fortuna, se ha descubierto que al dar a los pulsos cierta forma especial relacionada con el recíproco del coseno hiperbólico, casi todos los efectos de la dispersión se disipan y puede ser posible enviar pulsos a miles de kilómetros sin una distorsión apreciable de la forma. Estos pulsos se llaman **solitones.** Se está realizando un enorme esfuerzo de investigación para llevar a la práctica el uso de los solitones.

Cables de fibra

Los cables de fibra óptica son similares a los coaxiales, excepto por el trenzado. La figura 2-7(a) muestra una fibra individual vista de lado. Al centro se encuentra el núcleo de vidrio, a través del cual se propaga la luz. En las fibras multimodo el diámetro es de 50 micras, aproximadamente el grosor de un cabello humano. En las fibras monomodo el núcleo es de 8 a 10 micras.

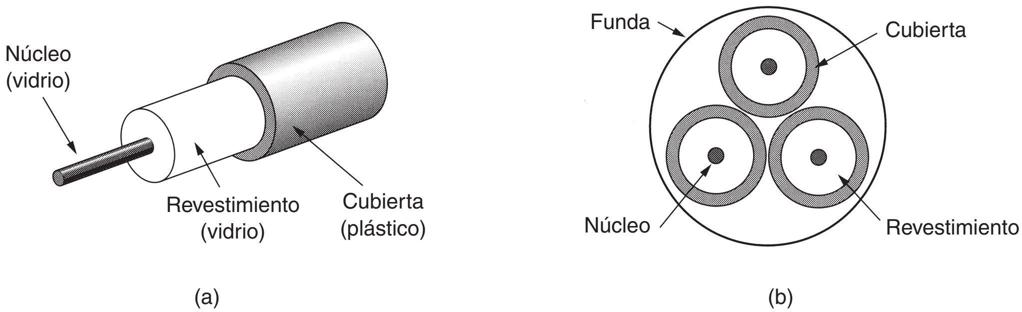


Figura 2-7. (a) Vista de lado de una fibra individual. (b) Vista de extremo de una funda con tres fibras.

El núcleo está rodeado por un revestimiento de vidrio con un índice de refracción menor que el del núcleo, con el fin de mantener toda la luz en este último. A continuación está una cubierta plástica delgada para proteger al revestimiento. Las fibras por lo general se agrupan en haces, protegidas por una funda exterior. La figura 2-7(b) muestra una funda con tres fibras.

Las cubiertas de fibras terrestres por lo general se colocan en el suelo a un metro de la superficie, donde a veces pueden sufrir daños ocasionados por retroexcavadoras o tuzas. Cerca de la costa, las cubiertas de fibras transoceánicas se entierran en zanjas mediante una especie de arado marino. En las aguas profundas, simplemente se colocan al fondo, donde los barcos de arrastre pueden tropezar con ellas o los calamares gigantes pueden atacarlas.

Las fibras se pueden conectar de tres formas diferentes. Primera, pueden terminar en conectores e insertarse en enchufes de fibra. Los conectores pierden entre 10 y 20% de la luz, pero facilitan la reconfiguración de los sistemas.

Segunda, se pueden empalmar de manera mecánica. Los empalmes mecánicos acomodan dos extremos cortados con cuidado, uno junto a otro, en una manga especial y los sujetan en su lugar. La alineación se puede mejorar pasando luz a través de la unión y haciendo pequeños ajustes para maximizar la señal. Personal especializado realiza los empalmes mecánicos en alrededor de cinco minutos, y la pérdida de luz de estos empalmes es de 10%.

Tercera, se pueden fusionar (fundir) dos tramos de fibra para formar una conexión sólida. Un empalme por fusión es casi tan bueno como una sola fibra, pero aun aquí hay un poco de atenuación.

Con los tres tipos de empalme pueden ocurrir reflejos en el punto del empalme, y la energía reflejada puede interferir la señal.

Por lo general se utilizan dos clases de fuente de luz para producir las señales: LEDs (diodos emisores de luz) y láseres semiconductores. Estas fuentes tienen propiedades diferentes, como se muestra en la figura 2-8, y su longitud de onda se puede ajustar mediante la inserción de interferómetros Fabry-Perot o Mach-Zehnder entre la fuente y la fibra. Los interferómetros Fabry-Perot son cavidades simples de resonancia que consisten en dos espejos paralelos. La luz incide de manera perpendicular en los espejos. La longitud de la cavidad separa las longitudes de onda que caben en ella un número entero de veces. Los interferómetros de Mach-Zehnder separan la luz en dos haces. Éstos viajan distancias ligeramente diferentes. Se vuelven a combinar en el extremo y quedan en fase sólo para ciertas longitudes de onda.

Elemento	LED	Láser semiconductor
Tasa de datos	Baja	Alta
Tipo de fibra	Multimodo	Multimodo o monomodo
Distancia	Corta	Larga
Tiempo de vida	Largo	Corto
Sensibilidad a la temperatura	Menor	Considerable
Costo	Bajo	Elevado

Figura 2-8. Comparación de diodos semiconductores y LEDs como fuentes de luz.

El extremo receptor de una fibra óptica consiste en un fotodiodo, el cual emite un pulso eléctrico cuando lo golpea la luz. El tiempo de respuesta típico de un fotodiodo es 1 nseg, lo que limita las tasas de datos a aproximadamente 1 Gbps. El ruido térmico también es un problema, por lo que un pulso de luz debe llevar suficiente potencia para que se pueda detectar. Al hacer que los pulsos tengan suficiente potencia, la tasa de errores puede disminuirse de manera considerable.

Redes de fibra óptica

La fibra óptica se puede utilizar en LANs, así como en transmisiones de largo alcance, aunque conectarse a ellas es más complicado que a una Ethernet. Una forma de superar el problema es reconocer que una red de anillo es en realidad una colección de enlaces punto a punto, como se muestra en la figura 2-9. La interfaz en cada computadora pasa el flujo de pulsos de luz hacia el siguiente enlace y también sirve como unión T para que la computadora pueda enviar y aceptar mensajes.

Se usan dos tipos de interfaz. Una interfaz pasiva consiste en dos derivaciones fusionadas a la fibra principal. Una derivación tiene un LED o un diodo láser en su extremo (para transmitir) y la otra tiene un fotodiodo (para recibir). La derivación misma es pasiva por completo y, por lo

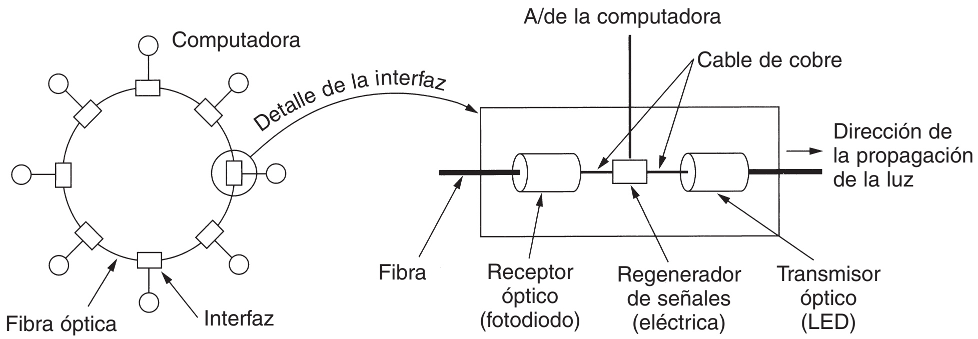


Figura 2-9. Anillo de fibra óptica con repetidores activos.

mismo, es extremadamente confiable pues un LED o un fotodiodo descompuesto no romperá el anillo, sólo dejará fuera de línea a una computadora.

El otro tipo de interfaz, mostrado en la figura 2-9, es el **repetidor activo.** La luz entrante se convierte en una señal eléctrica que se regenera a toda su intensidad si se debilitó y se retransmite como luz. La interfaz con la computadora es un alambre ordinario de cobre que entra en el regenerador de señales. En la actualidad también se usan los repetidores puramente ópticos. Estos dispositivos no requieren las conversiones óptica a eléctrica a óptica, lo que significa que pueden operar con anchos de banda muy altos.

Si falla un repetidor activo, el anillo se rompe y la red se cae. Por otro lado, puesto que la señal se regenera en cada interfaz, los enlaces individuales de computadora a computadora pueden tener una longitud de kilómetros, virtualmente sin un límite para el tamaño total del anillo. Las interfaces pasivas pierden luz en cada unión, de modo que la cantidad de computadoras y la longitud total del anillo se restringen en forma considerable.

La topología de anillo no es la única manera de construir una LAN con fibra óptica. También es posible tener difusión por hardware utilizando la construcción de **estrella pasiva** de la figura 2-10. En este diseño, cada interfaz tiene una fibra que corre desde su transmisor hasta un cilindro de sílice, con las fibras entrantes fusionadas a un extremo del cilindro. En forma similar, las fibras fusionadas al otro extremo del cilindro corren hacia cada uno de los receptores. Siempre que una interfaz emite un pulso de luz, se difunde dentro de la estrella pasiva para iluminar a todos los receptores, con lo que se alcanza la difusión. En efecto, la estrella pasiva combina todas las señales entrantes y transmite el resultado combinado por todas las líneas. Puesto que la energía entrante se divide entre todas las líneas que salen, la cantidad de nodos en la red está limitada por la sensibilidad de los fotodiodos.

Comparación de la fibra óptica y el alambre de cobre

Es instructivo comparar la fibra con el cobre. La fibra tiene muchas ventajas. Para empezar, puede manejar anchos de banda mucho mayores que el cobre. Tan sólo por esto, su uso sería indispensable en redes de alto rendimiento. Debido a la baja atenuación, sólo se necesitan repetidores cada 50 km aproximadamente en líneas largas, contra casi cada 5 km cuando se usa cobre, lo que

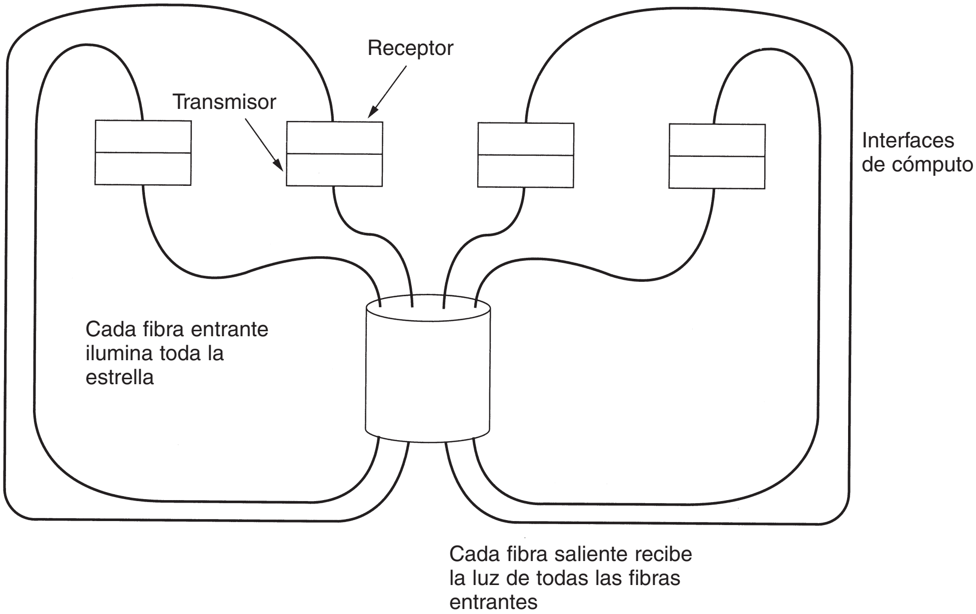


Figura 2-10. Conexión de estrella pasiva en una red de fibra óptica.

implica un ahorro considerable. La fibra también tiene la ventaja de que las sobrecargas de energía, la interferencia electromagnética o los cortes en el suministro de energía no la afectan. Las sustancias corrosivas del ambiente tampoco la afectan, lo que la hace ideal para ambientes fabriles pesados.

A las compañías telefónicas les gusta la fibra por una razón diferente: es delgada y ligera. Muchos conductos de cable existentes están completamente llenos, por lo que no hay espacio para agregar más capacidad. Al eliminar todo el cobre y reemplazarlo por fibra, se vacían los conductos y el cobre tiene un valor de reventa excelente para los refinadores de cobre quienes lo aprecian como materia prima de alta calidad. Además, las fibras son más ligeras que el cobre. Mil cables de par trenzado de 1 km pesan 8000 kg. Dos fibras tienen más capacidad y pesan sólo 100 kg, lo cual reduce de manera significativa la necesidad de sistemas mecánicos de apoyo que tienen que mantenerse. Para las nuevas rutas, la fibra se impone debido a su bajo costo de instalación.

Por último, las fibras no tienen fugas de luz y es difícil intervenirlas y conectarse a ellas. Estas propiedades dan a las fibras una seguridad excelente contra posibles espías.

Su parte negativa consiste en que es una tecnología poco familiar que requiere habilidades de las cuales carece la mayoría de los ingenieros, y en que las fibras pueden dañarse con facilidad si se doblan demasiado. Debido a que la transmisión óptica es unidireccional, la comunicación en ambos sentidos requiere ya sea dos fibras o dos bandas de frecuencia en una fibra. Por último, las interfaces de fibra cuestan más que las eléctricas. No obstante, el futuro de todas las comunicaciones fijas de datos para distancias de más de unos cuantos metros claramente es la fibra. Para un análisis de todos los aspectos de la fibra óptica y sus redes, vea (Hecht, 2001).

2.3 TRANSMISIÓN INALÁMBRICA

En nuestra era han surgido los adictos a la información: gente que necesita estar todo el tiempo en línea. Para estos usuarios móviles, el cable de par trenzado, el cable coaxial y la fibra óptica no son útiles. Ellos necesitan obtener datos para sus computadoras *laptop*, *notebook*, de bolsillo, de mano o de reloj pulsera sin estar limitados a la infraestructura de comunicaciones terrestre. Para estos usuarios, la comunicación inalámbrica es la respuesta. En las siguientes secciones veremos la comunicación inalámbrica en general, y veremos que tiene otras aplicaciones importantes además de proporcionar conectividad a los usuarios que desean navegar por Web desde la playa.

Algunas personas creen que en el futuro sólo habrá dos clases de comunicación: de fibra óptica e inalámbrica. Todos los aparatos fijos (es decir, no móviles): computadoras, teléfonos, faxes, etcétera, se conectarán con fibra óptica; todos los aparatos móviles usarán comunicación inalámbrica.

Sin embargo, la comunicación inalámbrica también tiene ventajas para los dispositivos fijos en ciertas circunstancias. Por ejemplo, si es difícil tender fibras hasta un edificio debido al terreno (montañas, selvas, pantanos, etcétera), podría ser preferible un sistema inalámbrico. Vale la pena mencionar que la comunicación digital inalámbrica moderna comenzó en las islas de Hawai, en donde partes considerablemente grandes del océano Pacífico separaban a los usuarios, y el sistema telefónico era inadecuado.

2.3.1 El espectro electromagnético

Cuando los electrones se mueven crean ondas electromagnéticas que se pueden propagar por el espacio libre (aun en el vacío). El físico británico James Clerk Maxwell predijo estas ondas en 1865 y el físico alemán Heinrich Hertz las observó en 1887. La cantidad de oscilaciones por segundo de una onda electromagnética es su **frecuencia,** f, y se mide en **Hz** (en honor a Heinrich Hertz). La distancia entre dos puntos máximos (o mínimos) consecutivos se llama **longitud de onda** y se designa de forma universal con la letra griega λ (lambda).

Al conectarse una antena del tamaño apropiado a un circuito eléctrico, las ondas electromagnéticas pueden ser difundidas de manera eficiente y ser captadas por un receptor a cierta distancia. Toda la comunicación inalámbrica se basa en este principio.

En el vacío, todas las ondas electromagnéticas viajan a la misma velocidad, no importa cuál sea su frecuencia. Esta velocidad, por lo general llamada **velocidad de la luz,** c, es de aproximadamente 3×10^8 m/seg o de un pie (30 cm) por nanosegundo. En el cobre o en la fibra óptica, la velocidad baja a casi 2/3 de este valor y se vuelve ligeramente dependiente de la frecuencia. La velocidad de la luz es el límite máximo de velocidad. Ningún objeto o señal puede moverse más rápido que la luz.

La relación fundamental entre f, λ y c (en el vacío) es:

$$\lambda f = c \tag{2-2}$$

Puesto que c es una constante, si conocemos el valor de f, podremos encontrar el de λ, y viceversa. Como regla general, cuando λ se expresa en metros y f en MHz, $\lambda f \approx 300$. Por ejemplo, las

ondas de 100 MHz son de aproximadamente 3 metros de longitud, las de 1000 MHz son de 0.3 metros y las ondas de 0.1 metros de longitud tienen una frecuencia de 3000 MHz.

En la figura 2-11 se muestra el espectro electromagnético. Las porciones de radio, microondas, infrarrojo y luz visible del espectro pueden servir para transmitir información modulando la amplitud, frecuencia o fase de las ondas. La luz ultravioleta, los rayos X y los rayos gamma serían todavía mejores, debido a sus frecuencias más altas, pero son difíciles de producir y modular, no se propagan bien entre edificios y son peligrosos para los seres vivos. Las bandas que se listan en la parte inferior de la figura 2-11 son los nombres oficiales de la ITU y se basan en las longitudes de onda, de modo que la banda LF va de 1 a 10 km (aproximadamente 30 a 300 kHz). Los términos LF, MF y HF se refieren a las frecuencias baja, media y alta, respectivamente. Como podrá observar, cuando se asignaron los nombres, nadie esperaba que se sobrepasarían los 10 MHz, por lo que posteriormente a las bandas más altas se les nombró como bandas VHF (frecuencia muy alta), UHF (frecuencia ultraalta), EHF (frecuencia extremadamente alta) y THF (frecuencia tremendamente alta). No hay más nombres aparte de éstos, pero IHF, AHF y PHF (increíblemente alta frecuencia, asombrosamente alta frecuencia y prodigiosamente alta frecuencia) sonarían bien.

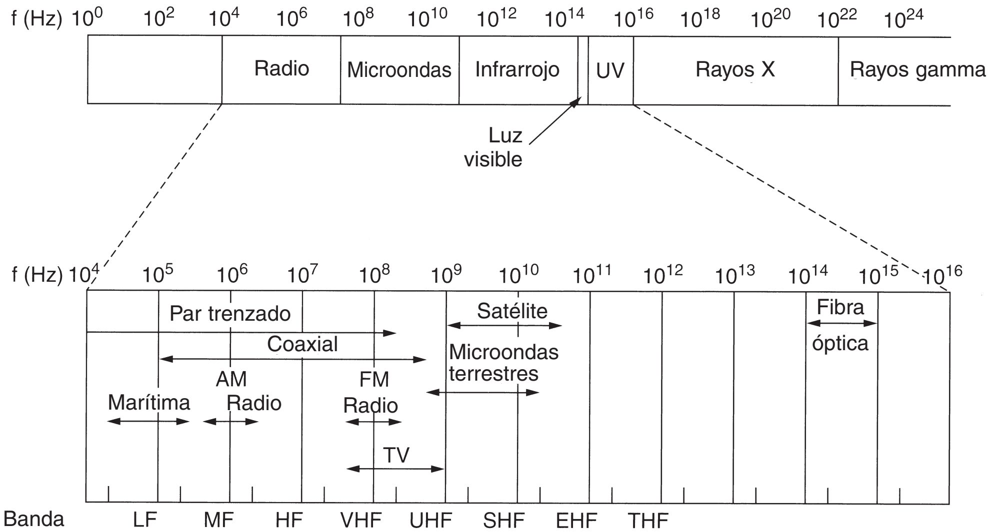


Figura 2-11. El espectro electromagnético y sus usos para comunicaciones.

La cantidad de información que puede transportar una onda electromagnética se relaciona con su ancho de banda. Con la tecnología actual, es posible codificar unos cuantos bits por hertz a frecuencias bajas, pero a frecuencias altas el número puede llegar hasta 8, de modo que un cable coaxial con un ancho de banda de 750 MHz puede transportar varios gigabits/seg. La figura 2-11 debe dejar en claro ahora por qué a la gente de redes le gusta tanto la fibra óptica.

Si resolvemos la ecuación (2-2) para f y la diferenciamos con respecto a λ, obtenemos

$$\frac{df}{d\lambda} = -\frac{c}{\lambda^2}$$

Si ahora usamos diferencias finitas en lugar de diferenciales y sólo consideramos los valores absolutos, obtenemos

$$\Delta f = \frac{c\Delta\lambda}{\lambda^2} \tag{2-3}$$

Por lo tanto, dado el ancho de una banda de longitud de onda, $\Delta\lambda$, podemos calcular la banda de frecuencia correspondiente, Δf, y a partir de ella, la tasa de datos que puede producir la banda. Cuanto más ancha sea ésta, mayor será la tasa de datos. Por ejemplo, considere la banda de 1.30 micras de la figura 2-6. Aquí tenemos $\lambda = 1.3 \times 10^{-6}$ y $\Delta\lambda = 0.17 \times 10^{-6}$, de manera que Δf es de aproximadamente 30 THz. A 8 bits/Hz, obtenemos 240 Tbps.

La mayoría de las transmisiones ocupa una banda de frecuencias estrecha (es decir, $\Delta f/f \ll 1$) a fin de obtener la mejor recepción (muchos watts/Hz). Sin embargo, en algunos casos se utiliza una banda ancha, con dos variaciones. En el **espectro disperso con salto de frecuencia,** el transmisor salta de frecuencia en frecuencia cientos de veces por segundo. Es popular en la comunicación militar debido a que de esta manera es difícil detectar las transmisiones y casi imposible intervenirlas. Ofrece buena resistencia al desvanecimiento por múltiples trayectorias debido a que la señal directa siempre llega primero al receptor. Las señales reflejadas siguen una trayectoria más larga y llegan más tarde. Para ese entonces, tal vez el receptor ya haya cambiado de frecuencia y no acepte señales de la frecuencia anterior, con lo que se elimina la interferencia entre las señales directas y reflejadas. En años recientes, esta técnica también se ha aplicado comercialmente —por ejemplo, tanto 802.11 como Bluetooth la utilizan.

Como nota curiosa, la atractiva austriaca Hedy Lamarr, la primera mujer que apareció desnuda en una película cinematográfica (el filme checoslovaco *Extase* de 1933), colaboró en la invención de esta técnica. Su primer esposo, un fabricante de armamento, le comentó lo fácil que era bloquear las señales de radio, las cuales en ese entonces se utilizaban para controlar los torpedos. Cuando descubrió que su esposo estaba vendiendo armas a Hitler, se horrorizó y se disfrazó de criada para escapar de él rumbo a Hollywood para continuar su carrera como actriz de cine. En su tiempo libre, inventó el salto de frecuencia para ayudar a los aliados en la guerra. Su diseño utilizaba 88 frecuencias, el número de teclas (y frecuencias) de un piano. Por su invento, ella y el compositor de música George Antheil, su amigo, recibieron la patente 2,292,387 de Estados Unidos. Sin embargo, no pudieron convencer a la Marina de Estados Unidos de que su invento era útil y, por lo tanto, nunca recibieron regalías. Años después de que la patente expirara, su invento cobró popularidad.

El otro tipo de espectro disperso, el **espectro disperso de secuencia directa** —el cual dispersa la señal a través una banda de frecuencia ancha—, está ganando popularidad en el mundo comercial. En particular, algunos teléfonos móviles de segunda generación lo utilizan, y dominará en los de tercera generación, gracias a su buena eficiencia espectral, inmunidad al ruido y otras propiedades. Algunas LANs inalámbricas también lo utilizan. Posteriormente volveremos al tema del espectro disperso. Si desea ver una historia fascinante y detallada de las comunicaciones por espectro disperso, vea (Scholtz, 1982).

Por el momento, supondremos que todas las transmisiones utilizan una banda de frecuencia estrecha. Ahora veremos cómo se emplean las distintas partes del espectro electromagnético de la figura 2-11, comenzando por la radio.

2.3.2 Radiotransmisión

Las ondas de radio son fáciles de generar, pueden viajar distancias largas y penetrar edificios sin problemas, y por ello su uso está muy generalizado en la comunicación, tanto en interiores como en exteriores. Las ondas de radio también son omnidireccionales, lo que significa que viajan en todas direcciones a partir de la fuente, por lo que no es necesario que el transmisor y el receptor se encuentren alineados físicamente.

En ocasiones la radio omnidireccional es buena, y en otras no lo es tanto. En la década de 1970, General Motors decidió equipar sus Cadillacs nuevos con frenos antibloqueo controlados por computadora. Cuando el conductor pisaba el pedal del freno, la computadora accionaba los frenos de manera intermitente en lugar de bloquearlos firmemente. Un buen día, un oficial que patrullaba las carreteras de Ohio encendió su nuevo radio móvil para llamar a su cuartel general y, de repente, el Cadillac que iba junto a él empezó a comportarse de manera muy extraña. El oficial le indicó al conductor que se detuviera a un lado del camino y, cuando lo hizo, el conductor alegó que él nada había hecho y que el carro se había vuelto loco.

Con el tiempo empezó a surgir un patrón: los Cadillacs ocasionalmente se comportaban de manera muy extraña, pero sólo en las principales carreteras de Ohio y sólo cuando alguna patrulla de caminos estaba cerca. Durante mucho tiempo General Motors no pudo comprender por qué los Cadillacs funcionaban bien en todos los demás estados e incluso en los caminos secundarios de Ohio. Después de una búsqueda intensa descubrieron que el cableado de los Cadillacs constituía una excelente antena para la frecuencia que usaba el nuevo sistema de radio de las patrullas de caminos de Ohio.

Las propiedades de las ondas de radio dependen de la frecuencia. A bajas frecuencias, esas ondas cruzan bien casi cualquier obstáculo, pero la potencia se reduce de manera drástica a medida que se aleja de la fuente, aproximadamente en proporción a $1/r^2$ en el aire. A frecuencias altas, las ondas de radio tienden a viajar en línea recta y a rebotar en los obstáculos. También son absorbidas por la lluvia. En todas las frecuencias, las ondas de radio están sujetas a interferencia por los motores y otros equipos eléctricos.

Por la capacidad del radio de viajar largas distancias, la interferencia entre usuarios es un problema. Por esta razón, todos los gobiernos reglamentan estrictamente el uso de radiotransmisores, con una excepción, que veremos más adelante.

En las bandas VLF, LF y MF las ondas de radio siguen la curvatura de la Tierra, como se ilustra en la figura 2-12(a). Estas ondas se pueden detectar quizá a 1000 km en las frecuencias más bajas, y a menos en frecuencias más altas. La difusión de radio AM usa la banda MF, y es por ello que las estaciones de radio AM de Boston no se pueden oír con facilidad en Nueva York. Las ondas de radio en estas bandas cruzan con facilidad los edificios, y es por ello que los radios portátiles funcionan en interiores. El problema principal al usar bandas para comunicación de datos es su ancho de banda bajo (vea la ecuación 2-3).

En las bandas HF y VHF, las ondas a nivel del suelo tienden a ser absorbidas por la tierra. Sin embargo, las ondas que alcanzan la ionosfera, una capa de partículas cargadas que rodea a la Tierra a una altura de 100 a 500 km, se refractan y se envían de regreso a nuestro planeta, como se muestra en la figura 2-12(b). En ciertas condiciones atmosféricas, las señales pueden rebotar varias veces. Los operadores de radio aficionados usan estas bandas para conversar a larga distancia. El ejército se comunica también en las bandas HF y VHF.

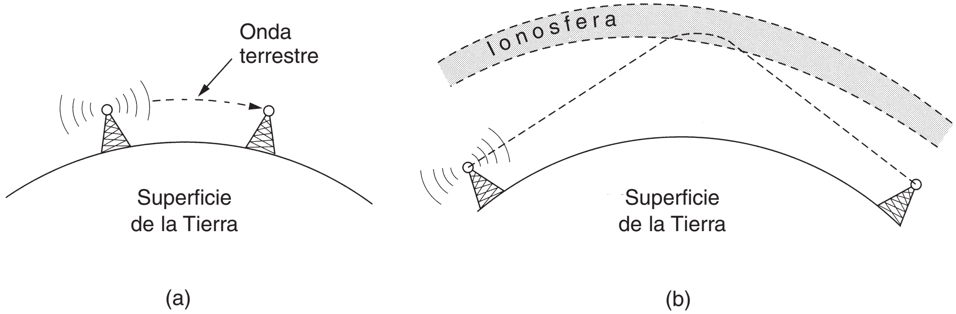


Figura 2-12. (a) En las bandas VLF, LF y MF, las ondas de radio siguen la curvatura de la Tierra. (b) En la banda HF las ondas rebotan en la ionosfera.

2.3.3 Transmisión por microondas

Por encima de los 100 MHz las ondas viajan en línea recta y, por lo tanto, se pueden enfocar en un haz estrecho. Concentrar toda la energía en un haz pequeño con una antena parabólica (como el tan familiar plato de televisión por satélite) produce una relación señal a ruido mucho más alta, pero las antenas transmisora y receptora deben estar bien alineadas entre sí. Además, esta direccionalidad permite que varios transmisores alineados en una fila se comuniquen sin interferencia con varios receptores en fila, siempre y cuando se sigan algunas reglas de espaciado. Antes de la fibra óptica, estas microondas formaron durante décadas el corazón del sistema de transmisión telefónica de larga distancia. De hecho, MCI, uno de los primeros competidores de AT&T después de que esta compañía fue desregularizada, construyó todo su sistema utilizando las comunicaciones mediante microondas que iban de torre en torre ubicadas a decenas de kilómetros una de la otra. Incluso el nombre de la compañía reflejó esto (MCI son las siglas de Microwave Communications, Inc.). Tiempo después, MCI adoptó la fibra y se fusionó con WorldCom.

Ya que las microondas viajan en línea recta, si las torres están muy separadas, partes de la Tierra estorbarán (piense en un enlace de San Francisco a Ámsterdam). Como consecuencia, se necesitan repetidores periódicos. Cuanto más altas sean las torres, más separadas pueden estar. La distancia entre los repetidores se eleva en forma muy aproximada con la raíz cuadrada de la altura de las torres. Con torres de 100 m de altura, los repetidores pueden estar separados a 80 km de distancia.

A diferencia de las ondas de radio a frecuencias más bajas, las microondas no atraviesan bien los edificios. Además, aun cuando el haz puede estar bien enfocado en el transmisor, hay cierta divergencia en el espacio. Algunas ondas pueden refractarse en las capas atmosféricas más bajas y tardar un poco más en llegar que las ondas directas. Las ondas diferidas pueden llegar fuera de fase con la onda directa y cancelar así la señal. Este efecto se llama **desvanecimiento por múltiples trayectorias** y con frecuencia es un problema serio que depende del clima y de la frecuencia. Algunos operadores mantienen 10% de sus canales inactivos como repuesto para activarlos cuando el desvanecimiento por múltiples trayectorias cancela en forma temporal alguna banda de frecuencia.

La creciente demanda de espectro obliga a los operadores a usar frecuencias más altas. Las bandas de hasta 10 GHz ahora son de uso rutinario, pero con las de aproximadamente 4 GHz surge un problema: son absorbidas por el agua. Estas ondas sólo tienen unos centímetros de longitud y la lluvia las absorbe. Este efecto sería útil si se quisiera construir un enorme horno de microondas externo para rostizar a los pájaros que pasen por ahí, pero para la comunicación es un problema grave. Al igual que con el desvanecimiento por múltiples trayectorias, la única solución es interrumpir los enlaces afectados por la lluvia y enrutar la comunicación por otra trayectoria.

En resumen, la comunicación por microondas se utiliza tanto para la comunicación telefónica de larga distancia, los teléfonos celulares, la distribución de la televisión, etcétera, que el espectro se ha vuelto muy escaso. Esta tecnología tiene varias ventajas significativas respecto a la fibra. La principal es que no se necesita derecho de paso; basta con comprar un terreno pequeño cada 50 km y construir en él una torre de microondas para saltarse el sistema telefónico y comunicarse en forma directa. Así es como MCI logró establecerse tan rápidamente como una compañía nueva telefónica de larga distancia. (Sprint siguió un camino totalmente diferente: la fundó el ferrocarril Southern Pacific Railroad, que ya poseía una gran cantidad de derechos de paso, limitándose a enterrar la fibra junto a las vías.)

Las microondas también son relativamente baratas. Erigir dos torres sencillas (podrían ser simplemente postes grandes con cables de retén) y poner antenas en cada una puede costar menos que enterrar 50 km de fibra a través de un área urbana congestionada o sobre una montaña, y también puede ser más económico que rentar la fibra de la compañía de teléfonos, en especial si ésta aún no ha recuperado por completo la inversión hecha por el cobre que quitó cuando instaló la fibra.

Las políticas del espectro electromagnético

Para evitar el caos total, hay acuerdos nacionales e internacionales acerca de quién utiliza cuáles frecuencias. Puesto que todos desean una tasa de transferencia de datos más alta, también desean más espectro. Los gobiernos nacionales asignan espectros para la radio AM y FM, la televisión y los teléfonos móviles, así como para las compañías telefónicas, la policía, la marina, la navegación, la milicia, el gobierno y muchos otros usuarios en competencia. A nivel mundial, una agencia de la ITU-R (WARC) trata de coordinar esta asignación de manera que se puedan fabricar los dispositivos que operan en diversos países. Sin embargo, los países no están atados a las recomendaciones de la ITU-R por lo que la FCC (Comisión Federal de Comunicaciones), que hace la asignación para Estados Unidos, ha rechazado ocasionalmente las recomendaciones de la ITU-R (por lo general, porque estas recomendaciones pedían a algún grupo políticamente poderoso que cediera una parte del espectro).

Incluso cuando una parte del espectro se ha asignado para un uso en particular, como para los teléfonos móviles, existe el aspecto adicional de cuál empresa portadora tiene permitido utilizar cuáles frecuencias. En el pasado se utilizaban tres algoritmos. El más antiguo, llamado **concurso de méritos** (*beauty contest*), requiere que cada empresa portadora explique por qué su propuesta es la que sirve mejor para los intereses públicos. Después los funcionarios del gobierno deciden

cuál de todas esas historias los convence más. Debido a que alguno de estos funcionarios otorgaban propiedad valuada en miles de millones de dólares a la compañía de su preferencia, esto conducía a soborno, corrupción, nepotismo, etcétera. Además, incluso un funcionario escrupulosamente honesto que piense que una compañía extranjera podría hacer mejor trabajo que cualquiera de las nacionales, tiene que dar muchas explicaciones.

Esta observación nos lleva al segundo algoritmo, en el que se realiza un sorteo entre las compañías interesadas. El problema con esta idea es que las compañías que no tienen ningún interés en utilizar el espectro, pueden entrar al sorteo. Por ejemplo, si un restaurante de comida rápida o una cadena de zapaterías gana, puede revender el espectro a una empresa portadora, sacando una ganancia inmensa y sin ningún riesgo.

La concesión de ganancias inesperadas a compañías atentas, aunque aleatorias, ha sido severamente criticada por muchos, lo que nos lleva al algoritmo 3: subastar el ancho de banda al mejor postor. Cuando en el año 2000 Inglaterra subastó las frecuencias necesarias para los sistemas móviles de la tercera generación, esperaba obtener aproximadamente $4 mil millones. En realidad recibió $40 mil millones debido a que las empresas portadoras cayeron en la desesperación ante la posibilidad de perder el mercado móvil. Este evento despertó la avaricia de los gobiernos vecinos y los motivó a llevar a cabo sus propias subastas. Funcionó, pero también dejó a algunas empresas portadoras con deudas enormes que ahora las tienen al borde de la bancarrota. Incluso en los mejores casos, les tomará muchos años recuperar la inversión en la licencia.

Un enfoque totalmente diferente para asignar frecuencias es no asignarlas por completo. Tan sólo se deja que todos transmitan a voluntad, pero se regula la potencia utilizada de manera que las estaciones tengan un rango tan corto que no interfieran entre ellas. Por consiguiente, la mayoría de los gobiernos han apartado algunas bandas de frecuencias, llamadas bandas **ISM** (**industriales, médicas y científicas**) de uso no autorizado. Los dispositivos para abrir puertas de garaje, teléfonos inalámbricos, juguetes controlados por radio, ratones inalámbricos y muchos otros dispositivos inalámbricos domésticos utilizan las bandas ISM. Para minimizar la interferencia entre estos dispositivos no coordinados, la FCC exige que todos los dispositivos que utilizan las bandas ISM utilicen técnicas de espectro disperso. En algunos otros países se aplican reglas similares.

La ubicación de las bandas ISM varía un poco de país a país. Por ejemplo, en Estados Unidos los dispositivos cuya potencia esté debajo de 1 watt, pueden utilizar las bandas que se muestran en la figura 2-13 sin requerir una licencia de la FCC. La banda de 900 MHz funciona mejor, pero está atestada y no está disponible en todo el mundo. La banda de 2.4 GHz está disponible en la mayoría de los países, pero está sujeta a interferencia por parte de los hornos de microondas e instalaciones de radar. Bluetooth y algunas de las LANs inalámbricas 802.11 operan en esta banda. La banda de 5.7 GHz es nueva y no se ha desarrollado del todo, por lo que el equipo que la utiliza es costoso, pero debido a que 802.11a la utiliza, se popularizará con rapidez.

2.3.4 Ondas infrarrojas y milimétricas

Las ondas infrarrojas y milimétricas no guiadas se usan mucho para la comunicación de corto alcance. Todos los controles remotos de los televisores, grabadoras de vídeo y estéreos utilizan comunicación infrarroja. Estos controles son relativamente direccionales, económicos y fáciles de

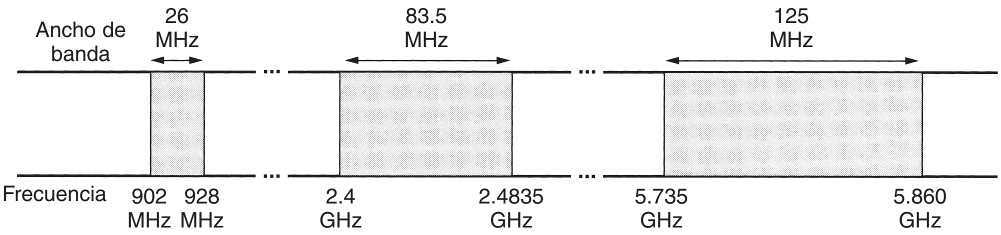


Figura 2-13. Las bandas ISM de Estados Unidos.

construir, pero tienen un inconveniente importante: no atraviesan los objetos sólidos (párese entre su televisor y su control remoto y vea si todavía funciona). En general, conforme pasamos de la radio de onda larga hacia la luz visible, las ondas se comportan cada vez más como la luz y cada vez menos como la radio.

Por otro lado, el hecho de que las ondas infrarrojas no atraviesen bien las paredes sólidas también es una ventaja. Esto significa que un sistema infrarrojo en un cuarto de un edificio no interferirá con un sistema similar en cuartos adyacentes. Por esta razón, la seguridad de estos sistemas contra el espionaje es mejor que la de los sistemas de radio. Además, no es necesario obtener licencia del gobierno para operar un sistema infrarrojo, en contraste con los sistemas de radio, que deben tener licencia afuera de las bandas ISM. La comunicación infrarroja tiene un uso limitado en el escritorio; por ejemplo, para conectar computadoras portátiles e impresoras, aunque no es un protagonista principal en el juego de la comunicación.

2.3.5 Transmisión por ondas de luz

La señalización óptica sin guías se ha utilizado durante siglos. Paul Revere utilizó señalización óptica binaria desde la Iglesia Old North justo antes de su famoso viaje. Una aplicación más moderna es conectar las LANs de dos edificios por medio de láseres montados en sus azoteas. La señalización óptica coherente con láseres es inherentemente unidireccional, de modo que cada edificio necesita su propio láser y su propio fotodetector. Este esquema ofrece un ancho de banda muy alto y un costo muy bajo. También es relativamente fácil de instalar y, a diferencia de las microondas, no requiere una licencia de la FCC.

Sin embargo, la ventaja del láser, un haz muy estrecho, aquí también es una debilidad. Apuntar un rayo láser de 1 mm de anchura a un blanco del tamaño de la punta de un alfiler a 500 m de distancia requiere la puntería de una Annie Oakley moderna. Por lo general, se añaden lentes al sistema para desenfocar ligeramente el rayo.

Una desventaja es que los rayos láser no pueden penetrar la lluvia ni la niebla densa, pero normalmente funcionan bien en días soleados. Sin embargo, en una ocasión el autor asistió a una conferencia en un moderno hotel de Europa en el que los organizadores tuvieron la atención de proporcionar un salón lleno de terminales para que los asistentes leyeran su correo electrónico durante las presentaciones aburridas. Puesto que la PTT local no estaba dispuesta a instalar un gran

número de líneas telefónicas sólo para tres días, los organizadores colocaron un láser en el techo, lo apuntaron al edificio de ciencias de la computación de su universidad, el cual está a unos cuantos kilómetros de allí; lo probaron la noche anterior a la conferencia y funcionó a la perfección. A las 9 a.m. del siguiente día, que era brillante y soleado, el enlace falló por completo y permaneció caído todo el día. Esa noche los organizadores volvieron a probar con mucho cuidado el enlace y de nuevo funcionó a la perfección. El patrón se repitió durante dos días más de forma idéntica.

Después de la conferencia, los organizadores descubrieron el problema. Durante el día, el calor del sol causaba corrientes de convección que se elevaban desde el techo del edificio, como se muestra en la figura 2-14. Este aire turbulento desviaba el rayo y lo hacía danzar alrededor del detector. Una “vista” atmosférica como ésta hace titilar a las estrellas (y es la razón por la cual los astrónomos ponen sus telescopios en las cimas de las montañas, para quedar tan arriba en la atmósfera como sea posible). Este fenómeno es también la causa del aspecto trémulo de las carreteras en un día caluroso y de las imágenes ondulantes cuando se mira sobre un radiador caliente.

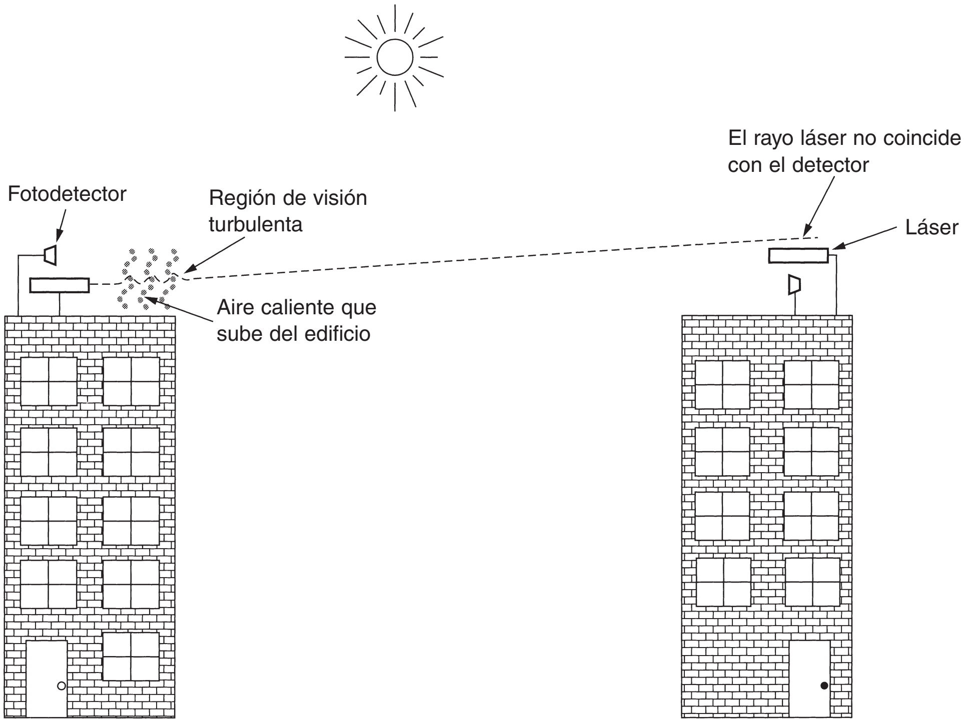


Figura 2-14. Las corrientes de convección pueden interferir los sistemas de comunicación por láser. Aquí se ilustra un sistema bidireccional con dos láseres.

2.4 SATÉLITES DE COMUNICACIONES

En la década de 1950 y principios de la de 1960, hubo intentos por establecer sistemas de comunicación mediante el rebote de señales sobre globos climáticos. Por desgracia, las señales que se recibían eran demasiado débiles para darles un uso práctico. Entonces, la Marina de Estados Unidos descubrió una especie de globo climático en el cielo —la Luna— y desarrolló un sistema de comunicaciones por repetición (o de barco a tierra) que rebotaba señales de él.

Progresos posteriores en el campo de las comunicaciones por el cielo tuvieron que esperar hasta que se lanzó el primer satélite de comunicaciones. La principal diferencia entre un satélite artificial y uno real es que el primero puede amplificar las señales antes de mandarlas de regreso, convirtiendo lo que parecía una idea estrafalaria en un poderoso sistema de comunicaciones.

Los satélites de comunicaciones tienen algunas propiedades interesantes que los hacen atractivos para muchas aplicaciones. En su forma más simple, un satélite de comunicaciones se puede considerar como un enorme repetidor de microondas en el cielo. Contiene numerosos **transpondedores,** cada uno de los cuales se encarga de una parte del espectro, amplifica la señal entrante y a continuación la retransmite en otra frecuencia para evitar interferencia con la señal entrante. Los haces pueden ser amplios y cubrir una fracción sustancial de la superficie de la Tierra, o estrechos, y abarcar sólo algunos cientos de kilómetros de diámetro. Este modo de operación se conoce como de **tubo doblado.**

De acuerdo con la ley de Kepler, el periodo orbital de un satélite varía según el radio de la órbita a la 3/2 potencia. Entre más alto esté el satélite, más largo es el periodo. Cerca de la superficie de la Tierra, el periodo es de aproximadamente 90 minutos. En consecuencia, los satélites con órbitas bajas desaparecen de la vista con bastante rapidez, aunque algunos de ellos son necesarios para proporcionar una cobertura continua. A una altitud de cerca de 35,800 km, el periodo es de 24 horas. A una de 384,000 km, el periodo es cercano a un mes, como puede atestiguar cualquiera que haya observado la Luna con regularidad.

El periodo de un satélite es importante, aunque no es el único punto para determinar dónde colocarlo. Otro aspecto es la presencia de los cinturones de Van Allen, capas de partículas altamente cargadas de energía, atrapadas por el campo magnético de la Tierra. Cualquier satélite que vuele dentro de ellas sería destruido rápidamente por las partículas con una alta carga de energía. Del análisis de estos factores resulta que hay tres regiones para colocar con seguridad los satélites. En la figura 2-15 se muestran estas regiones y algunas de sus propiedades. Enseguida describiremos brevemente los satélites que habitan cada una de estas regiones.

2.4.1 Satélites geoestacionarios

En 1945, el escritor de ciencia-ficción Arthur C. Clarke calculó que un satélite a una altitud de 35,800 km en una órbita ecuatorial circular aparentaría permanecer inmóvil en el cielo, por lo que no sería necesario rastrearlo (Clarke, 1945). Se dio a la tarea de describir un sistema de comunicaciones completo que utilizaba estos (tripulados) **satélites geoestacionarios,** incluyendo

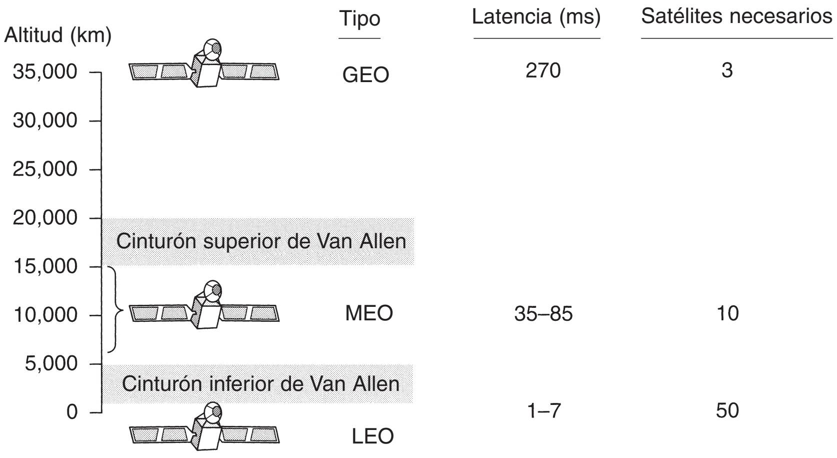


Figura 2-15. Satélites de comunicaciones y algunas de sus propiedades, entre ellas: altitud sobre la Tierra, tiempo de duración de un viaje de ida y vuelta y la cantidad de satélites necesarios para abarcar toda la Tierra.

las órbitas, paneles solares, radiofrecuencias y procedimientos de lanzamiento. Desafortunadamente, llegó a la conclusión de que los satélites no eran prácticos debido a la imposibilidad de poner en órbita amplificadores de tubos catódicos frágiles que consumían una gran cantidad de energía, por lo cual nunca le dio seguimiento a esta idea, aunque escribió algunos relatos de ciencia ficción al respecto.

La invención del transistor cambió las cosas, y el primer satélite de comunicaciones artificial, Telstar, fue lanzado en julio de 1962. Desde entonces, los satélites de comunicaciones se han convertido en un negocio multimillonario y en el único aspecto del espacio exterior altamente rentable. Con frecuencia, a estos satélites que vuelan a grandes alturas se les llama satélites **GEO** (**Órbita Terrestre Geoestacionaria**).

Con la tecnología actual, es poco aconsejable utilizar satélites geoestacionarios espaciados a menos de dos grados en el plano ecuatorial de 360 grados para evitar interferencia. Con un espaciamiento de dos grados, sólo puede haber 360/2 = 180 de estos satélites a la vez en el cielo. Sin embargo, cada transpondedor puede utilizar múltiples frecuencias y polarizaciones para incrementar el ancho de banda disponible.

Para evitar el caos total en el cielo, la ITU asigna la posición orbital. Este proceso tiene fuertes connotaciones políticas, y países que apenas están saliendo de la edad de piedra demandan "sus" posiciones orbitales (con el propósito de alquilarlas al mejor postor). No obstante, algunos países sostienen que la propiedad no se extiende a la Luna y que ningún país tiene derechos legales sobre las posiciones orbitales que se encuentran arriba de su territorio. Por si esto no fuera suficiente, las telecomunicaciones comerciales no son la única aplicación. Las compañías televisoras, los gobiernos y la milicia también quieren su tajada del pastel orbital.

Los satélites modernos pueden ser bastante grandes, pesar hasta 4000 kg y consumir varios kilowatts de electricidad producida por paneles solares. La gravedad del Sol, la Luna y los planetas

tiende a desplazar a los satélites de sus órbitas y orientaciones asignadas, efecto contrarrestado por los motores turbo integrados de los satélites. Esta actividad de ajuste se conoce como **control de la posición orbital.** Sin embargo, cuando se termina el combustible de los motores, por lo general a los 10 años, el satélite navega a la deriva y cae sin remedio, por lo cual es necesario desactivarlo. Con el tiempo, la órbita se deteriora y el satélite reingresa a la atmósfera y se incendia o en ocasiones se estrella contra la Tierra.

Las posiciones orbitales no son la única manzana de la discordia. También lo son las frecuencias, debido a que las transmisiones de los enlaces descendentes interfieren con los usuarios de microondas existentes. En consecuencia, la ITU ha asignado bandas de frecuencia específicas a los usuarios de satélites. Las principales se muestran en la figura 2-16. La banda C fue la primera que se destinó al tráfico comercial por satélite. Tiene dos rangos de frecuencia, el inferior para el tráfico descendente o de bajada (proveniente del satélite) y el superior para el tráfico ascendente o de subida (hacia el satélite). Para permitir que el tráfico fluya en ambos sentidos al mismo tiempo, se requieren dos canales, uno para cada sentido. Estas bandas están sobresaturadas debido a que las empresas portadoras también las utilizan para los enlaces de microondas terrestres. Las bandas L y S fueron incorporadas en el año 2000 mediante un acuerdo internacional. No obstante, son estrechas y saturadas.

Banda	Enlace descendente	Enlace ascendente	Ancho de banda	Problemas
L	1.5 GHz	1.6 GHz	15 MHz	Bajo ancho de banda; saturada
S	1.9 GHz	2.2 GHz	70 MHz	Bajo ancho de banda; saturada
C	4.0 GHz	6.0 GHz	500 MHz	Interferencia terrestre
Ku	11 GHz	14 GHz	500 MHz	Lluvia
Ka	20 GHz	30 GHz	3500 MHz	Lluvia, costo del equipo

Figura 2-16. Principales bandas de satélite.

La siguiente banda más ancha disponible para los operadores de telecomunicaciones es la banda Ku (K abajo). Esta banda aún no está saturada, y a estas frecuencias es posible espaciar los satélites a cerca de un grado. No obstante, hay otro problema: la lluvia. El agua es un gran absorbente de estas microondas cortas. La buena noticia es que por lo general las tormentas se confinan a sitios específicos, por lo que el problema se soluciona con la instalación de varias estaciones terrestres con suficiente separación en vez de una sola, al costo de más antenas, cables y aparatos electrónicos que permitan pasar rápidamente de una estación a otra. También se ha asignado ancho de banda para tráfico comercial por satélite en la banda Ka (K arriba), pero el equipo necesario para utilizar esta banda aún es caro. Además de estas bandas comerciales, también hay muchas bandas gubernamentales y militares.

Un satélite moderno tiene alrededor de 40 transpondedores, cada uno con un ancho de banda de 80 MHz. Por lo general, cada transpondedor opera como un tubo doblado, pero algunos satélites recientes tienen capacidad de procesamiento a bordo, lo cual les permite una operación más refinada. La división de los transpondedores en canales era estática en los primeros satélites: el

ancho de banda se dividía simplemente en bandas de frecuencia fija. En nuestros días, cada haz del transpondedor se divide en ranuras temporales, y varios usuarios su turnan para utilizarlo. Más tarde en este mismo capítulo analizaremos en detalle estas dos técnicas (multiplexión por división de frecuencia y multiplexión por división de tiempo).

Los primeros satélites geoestacionarios tenían un solo haz espacial que iluminaba cerca de un tercio de la superficie de la Tierra, al cual se le conoce como **huella.** Con la considerable reducción en precio, tamaño y requerimientos de energía de los componentes microelectrónicos, se ha vuelto posible una estrategia de difusión mucho más refinada. Cada satélite está equipado con múltiples antenas y transpondedores. Cada haz descendente se puede concentrar en un área geográfica pequeña, de tal forma que es posible llevar a cabo simultáneamente una gran cantidad de transmisiones hacia y desde el satélite. Normalmente, estos haces, conocidos como **haces reducidos,** tienen forma elíptica y pueden ser tan pequeños como algunos cientos de kilómetros. Por lo general, un satélite de comunicaciones para los Estados Unidos de América tiene un haz ancho para los 48 estados contiguos y haces reducidos para Alaska y Hawaii.

Un avance reciente en el mundo de los satélites de comunicaciones es el desarrollo de microestaciones de bajo costo, llamadas **VSATs** (**Terminales de Apertura Muy Pequeña**) (Abramson, 2000). Estas diminutas terminales tienen antenas de un metro o más pequeñas (en comparación con los 10 metros que mide una antena GEO estándar) y pueden producir alrededor de un watt de energía. Por lo general, el enlace ascendente funciona a 19.2 kbps, pero el enlace descendente funciona con frecuencia a 512 kbps o más. La televisión de difusión directa por satélite utiliza esta tecnología para transmisión unidireccional.

En muchos sistemas VSAT, las microestaciones no tienen suficiente potencia para comunicarse directamente una con la otra (a través del satélite, por supuesto). En vez de ello, como se muestra en la figura 2-17, es necesaria una estación especial en tierra, la **estación central,** que cuenta con una antena grande, para retransmitir el tráfico entre VSATs. En este modo de operación, el emisor o el receptor tienen una antena grande y un amplificador potente. La desventaja es que existe un retardo más prolongado al contar con estaciones de usuario más económicas.

Las VSATs tienen un futuro prometedor en las zonas rurales. Aún no tienen una amplia aceptación, pero más de la mitad de la población del mundo vive a una hora de distancia del teléfono más cercano. El tendido de redes telefónicas a miles de pequeñas poblaciones excede el presupuesto de la mayoría de los gobiernos del tercer mundo, pero lo que sí es factible es la instalación de antenas VSAT de un metro alimentadas por celdas solares. Las VSATs proporcionarán la tecnología que enlazará al mundo.

Los satélites de comunicaciones tienen diversas propiedades radicalmente distintas a las de los enlaces terrestres de punto a punto. Para empezar, aun cuando las señales hacia y desde un satélite viajan a la velocidad de la luz (cerca de 300,000 km/seg), el largo viaje de ida y vuelta provoca un retardo sustancial para los satélites GEO. Dependiendo de la distancia entre el usuario y la estación terrestre, así como de la elevación del satélite en el horizonte, el tiempo de tránsito de un extremo al otro es de entre 250 y 300 mseg. Un valor común es de 270 mseg (540 mseg para un sistema VSAT con una estación central).

Con propósitos de comparación, los enlaces terrestres de microondas tienen un retardo de propagación de casi 3 μseg/km, en tanto que los enlaces de cable coaxial o la fibra óptica tienen un

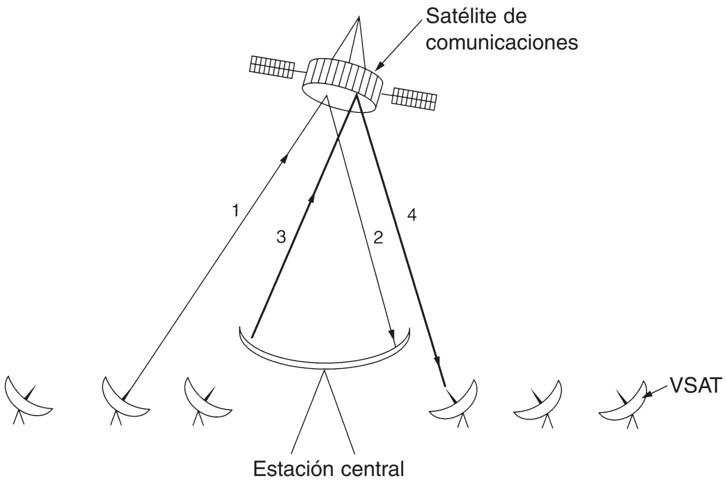


Figura 2-17. VSATs con una estación central.

retardo de aproximadamente 5 μseg/km. El último es más lento que el primero debido a que las señales electromagnéticas viajan más rápido en el aire que en materiales sólidos.

Otra propiedad importante de los satélites es que son esencialmente medios de difusión. No cuesta más enviar un mensaje a miles de estaciones dentro de la huella de un transpondedor de lo que cuesta enviarlo a una sola estación. Esta propiedad es muy útil para algunas aplicaciones. Por ejemplo, es posible que un satélite difunda páginas Web populares a los cachés de una gran cantidad de computadoras diseminadas en un área amplia. Aun cuando la difusión se puede simular con líneas punto a punto, la difusión por satélite es mucho más económica. Por otro lado, los satélites son un verdadero desastre en el aspecto de seguridad y privacidad: cualquiera puede escuchar todo. La encriptación es esencial cuando se requiere seguridad.

Los satélites también tienen la propiedad de que el costo de transmitir un mensaje es independiente de la distancia que se recorra. Una llamada al otro lado del océano tiene el mismo costo que una al otro lado de la calle. Los satélites también cuentan con excelentes tasas de error y se pueden desplegar de manera casi instantánea, un aspecto importante para las comunicaciones militares.

2.4.2 Satélites de Órbita Terrestre Media

Los satélites **MEO** (**Órbita Terrestre Media**) se encuentran a altitudes mucho más bajas, entre los dos cinturones de Van Allen. Vistos desde la Tierra, estos satélites se desplazan lentamente y tardan alrededor de seis horas para dar la vuelta a la Tierra. Por consiguiente, es necesario rastrearlos conforme se desplazan. Puesto que son menores que los GEO, tienen una huella más pequeña y se requieren transmisores menos potentes para alcanzarlos. Hoy en día no se utilizan para telecomunicaciones, por lo cual no los examinaremos aquí. Los 24 satélites **GPS** (**Sistema de Posicionamiento Global**) que orbitan a cerca de 18,000 km son ejemplos de satélites MEO.

2.4.3 Satélites de Órbita Terrestre Baja

En una altitud más baja encontramos a los satélites **LEO** (**Órbita Terrestre Baja**)**.** Debido a la rapidez de su movimiento, se requieren grandes cantidades de ellos para conformar un sistema completo. Por otro lado, como los satélites se encuentran tan cercanos a la Tierra, las estaciones terrestres no necesitan mucha potencia, y el retardo del viaje de ida y vuelta es de tan sólo algunos milisegundos. En esta sección examinaremos tres ejemplos, dos sobre las comunicaciones de voz y uno sobre el servicio de Internet.

Iridium

Como ya mencionamos, durante los primeros 30 años de la era de los satélites casi no se utilizaban los satélites de órbita baja porque aparecían y desaparecían con mucha rapidez. En 1990, Motorola abrió un nuevo camino al solicitar permiso a la FCC (Comisión Federal de Comunicaciones de Estados Unidos) para lanzar 77 satélites de órbita baja para el proyecto Iridium (el iridio es el elemento 77). El plan fue modificado más tarde para utilizar sólo 66 satélites, por lo que el proyecto debió haberse renombrado como Dysprosium (elemento 66), pero quizá este nombre sonaba demasiado a enfermedad. El propósito era que tan pronto como un satélite se perdiera de vista, otro lo reemplazaría. Esta propuesta desató un frenesí entre otras compañías. De pronto, todos querían lanzar una cadena de satélites de órbita baja.

Después de siete años de improvisación de socios y financiamiento, los socios lanzaron los satélites Iridium en 1997. El servicio de comunicaciones empezó en noviembre de 1998. Por desgracia, la demanda comercial de teléfonos por satélite grandes y pesados fue insignificante porque la red telefónica móvil había crecido de manera espectacular desde 1990. En consecuencia, el proyecto Iridium no fue rentable y quebró en agosto de 1999 en lo que fue uno de los fracasos corporativos más espectaculares de la historia. Los satélites y otros activos (con valor de 5000 millones de dólares) fueron adquiridos posteriormente por un inversionista en 25 millones de dólares en una especie de venta de garaje extraterrestre. El servicio Iridium se reinició en marzo de 2001.

El negocio de Iridium era (y es) ofrecer servicio de telecomunicaciones en todo el mundo a través de dispositivos de bolsillo que se comunican directamente con los satélites Iridium. Proporciona servicio de voz, datos, búsqueda de personas, fax y navegación en cualquier parte, sea en tierra, mar y aire. Entre sus clientes están las industrias marítima, de la aviación y exploración petrolera, así como personas que viajan a partes del mundo que carecen de infraestructura de telecomunicaciones (por ejemplo, desiertos, montañas, selvas y algunos países del tercer mundo).

Los satélites Iridium están a una altitud de 750 km, en órbitas polares circulares. Están dispuestos en forma de collar de norte a sur, con un satélite a cada 32 grados de latitud. La Tierra completa se cubre con seis collares, como se aprecia en la figura 2-18(a). Quienes no tengan muchos conocimientos sobre química pueden pensar que esta disposición es un gran átomo de disprosio, con la Tierra como núcleo y los satélites como electrones.

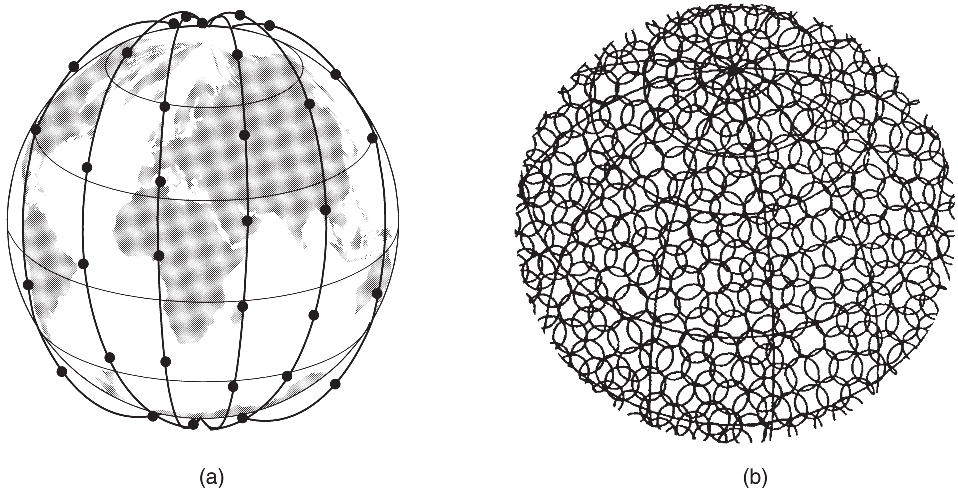


Figura 2-18. (a) Los satélites Iridium forman seis collares alrededor de la Tierra. (b) 1628 celdas en movimiento cubren la Tierra.

Cada satélite tiene un máximo de 48 celdas (haces reducidos), con un total de 1628 celdas sobre la superficie de la Tierra, como se muestra en la figura 2-18(b). Cada satélite tiene una capacidad de 3840 canales, o 253,440 en total. Algunos de estos canales se utilizan para localización de personas y navegación, en tanto que otros, para datos y voz.

Una propiedad interesante de Iridium es que la comunicación entre clientes distantes tiene lugar en el espacio, con un satélite retransmitiendo datos al siguiente, como se muestra en la figura 2-19(a). Aquí vemos que quien llama está en el Polo Norte y hace contacto con un satélite que se encuentra directamente arriba de él. La llamada se retransmite a través de otros satélites y por último es entregada al destinatario en el Polo Sur.

Globalstar

Globalstar es un diseño alterno para Iridium. Se basa en 48 satélites LEO pero utiliza un esquema de conmutación diferente al de Iridium. En tanto que Iridium retransmite las llamadas de satélite en satélite, lo cual requiere un equipo de conmutación refinado en los satélites, Globalstar utiliza un diseño de tubo doblado tradicional. La llamada que se originó en el Polo Norte en la figura 2-19(b) es devuelta a la Tierra y recogida por la enorme estación terrestre. A continuación la llamada se enruta, a través de una red terrestre, a la estación terrestre más cercana al destinatario y se entrega mediante una conexión de tubo doblado como se muestra. La ventaja de este esquema es que mucha de la complejidad queda en tierra, donde es más sencillo manejarla. Asimismo,

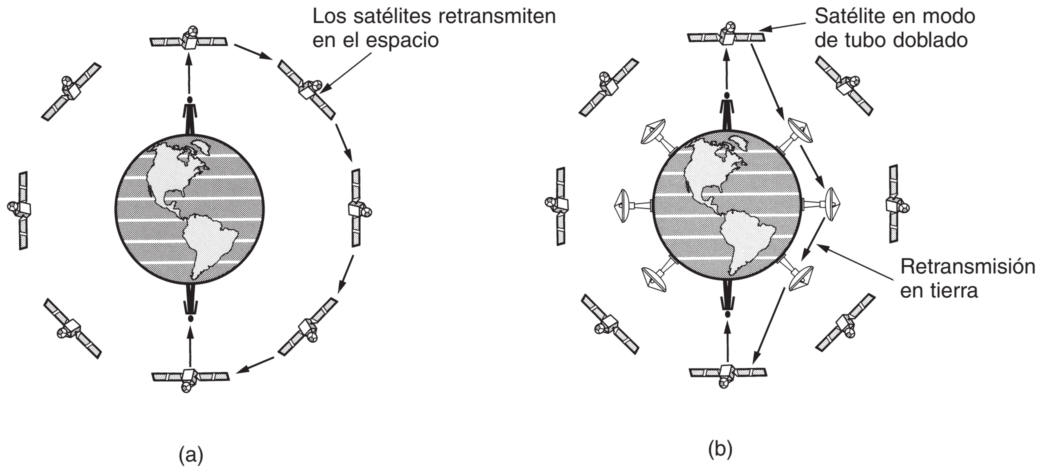


Figura 2-19. (a) Retransmisión en el espacio. (b) Retransmisión en tierra.

el uso de antenas grandes en las estaciones terrestres que pueden producir una señal potente y recibir una señal débil, permite la utilización de teléfonos de baja potencia. Después de todo, el teléfono produce tan sólo unos cuantos miliwatts de potencia, por lo cual la señal que llega a las estaciones terrestres es sumamente débil, aun cuando el satélite la haya amplificado.

Teledesic

Iridium está destinada a usuarios de teléfonos que se encuentran en lugares extremosos. Nuestro siguiente ejemplo, **Teledesic,** está destinada a usuarios de Internet de todo el mundo deseosos de ancho de banda. Fue concebida en 1990 por Craig McCaw, pionero de la telefonía móvil, y por Bill Gates, fundador de Microsoft, quienes estaban inconformes con el lento ritmo al cual las compañías telefónicas de todo el mundo proporcionaban ancho de banda alto a los usuarios de computadoras. La meta del sistema Teledesic es ofrecer a los millones de usuarios concurrentes de Internet un enlace ascendente de hasta 100 Mbps y un enlace descendente de hasta 720 Mbps mediante antenas tipo VSAT pequeñas y fijas, que ignoran por completo el sistema telefónico. Para las compañías telefónicas esto es demasiado bello para ser realidad.

El diseño original consistía en un sistema de 288 satélites de huella pequeña, dispuestos en 12 planos justo debajo del cinturón inferior de Van Allen a una altitud de 1350 km. Posteriormente se modificó el diseño a 30 satélites con huellas más grandes. La transmisión se realiza en la banda Ka, relativamente poco saturada y con un ancho de banda alto. El sistema es de conmutación de paquetes en el espacio, en el cual cada satélite tiene la capacidad de enrutar paquetes a los satélites vecinos. Cuando un usuario necesita ancho de banda para enviar paquetes, tal ancho de banda se solicita y asigna de manera dinámica en alrededor de 50 mseg. Si todo marcha bien, el sistema está programado para empezar a funcionar en 2005.

2.4.4 Satélites en comparación con fibra óptica

Una comparación entre comunicación por satélite y comunicación terrestre es aleccionadora. Apenas hace 20 años se podía afirmar que el futuro de las comunicaciones estaba en los satélites. Después de todo, el sistema telefónico ha cambiado poco en los pasados 100 años y no hay señales de que cambie en los próximos 100 años. Este lento movimiento fue ocasionado en gran parte por las regulaciones que obligaban a las compañías telefónicas a ofrecer un buen servicio de voz a precios razonables (lo cual hicieron), y a cambio obtuvieron utilidades garantizadas sobre sus inversiones. Para quienes tenían que transmitir datos, había módems de 1200 bps. Por mucho, esto es todo lo que había.

Esta situación cambió radicalmente en 1984 con la entrada de la competencia en Estados Unidos y un poco más tarde en Europa. Las compañías telefónicas comenzaron a reemplazar sus viejas redes con fibra óptica e introdujeron servicios de ancho de banda alto como ADSL (Línea Digital de Suscriptor Asimétrica). También suspendieron su añeja práctica de cargar precios artificialmente altos a los usuarios de larga distancia para subsidiar el servicio local.

De buenas a primeras, las conexiones terrestres de fibra óptica dieron la impresión de que serían las ganadoras a largo plazo. No obstante, los satélites de comunicaciones tienen algunos nichos de mercado importantes a los cuales la fibra óptica no se dirige (en ocasiones porque no puede). A continuación veremos algunos de ellos.

En primer lugar, a pesar de que una fibra óptica tiene más ancho de banda potencial que todos los satélites que se han lanzado, este ancho de banda no está disponible para la mayoría de los usuarios. La fibra que se instala actualmente se utiliza en el sistema telefónico para manejar muchas llamadas de larga distancia al mismo tiempo, no para ofrecer un ancho de banda alto a los usuarios individuales. Con los satélites, es factible que un usuario instale una antena en el techo de la casa y evada por completo el sistema telefónico para conseguir un ancho de banda alto. Teledesic se apoya en esta idea.

Un segundo nicho es el de la comunicación móvil. En estos días mucha gente desea comunicarse mientras trota, maneja, navega o vuela. Los enlaces terrestres de fibra óptica no sirven para este uso, pero los enlaces por satélite sí. Sin embargo, es posible que una combinación de radio celular y fibra óptica funcionara para la mayoría de los casos (aunque quizá no para aquellos que viajen por aire o por mar).

Un tercer nicho es para aquellas situaciones en las cuales se requiere difusión. Un mensaje enviado desde un satélite se puede recibir en miles de estaciones terrestres al mismo tiempo. Por ejemplo, para una organización que transmita un flujo de precios de acciones, bonos o commodities a miles de operadores de bolsa le resultaría más económico un sistema por satélite que simular la difusión en tierra.

Un cuarto nicho es el de las comunicaciones en lugares agrestes o con una infraestructura terrestre pobremente desarrollada. Por ejemplo, Indonesia tiene su propio satélite para el tráfico telefónico interno. El lanzamiento de un satélite resultó más económico que el enlace de miles de cables bajo el mar entre las 13,667 islas que conforman el archipiélago.

Un quinto nicho de mercado para los satélites son las áreas donde es difícil o extremadamente costoso conseguir un derecho para el tendido de fibra óptica.

Sexto, cuando un despliegue rápido es primordial, como en un sistema de comunicaciones militar en época de guerra, los satélites ganan con facilidad.

En resumen, al parecer la tendencia general de las comunicaciones en el futuro será la fibra óptica terrestre en combinación con radio celular, pero los satélites son mejores para algunos usos especializados. Sin embargo, hay un imponderable que se aplica en todos los casos: el aspecto económico. Aunque la fibra óptica ofrece más ancho de banda, es muy probable que las comunicaciones terrestres y por satélite competirán agresivamente en precio. Si los avances tecnológicos reducen de manera drástica el costo de despliegue de un satélite (por ejemplo, algún transbordador espacial futuro que pueda diseminar docenas de satélites en un solo lanzamiento) o los satélites de órbita baja se popularizan, no hay certeza de que la fibra óptica ganará en todos los mercados.

2.5 LA RED TELEFÓNICA PÚBLICA CONMUTADA

Cuando dos computadoras propiedad de la misma empresa u organización, localizadas cerca una de la otra, necesitan comunicarse, es más fácil conectarlas mediante un cable. Las LANs funcionan de esta manera. Sin embargo, cuando las distancias son considerables o hay muchas computadoras o los cables tienen que pasar por una vía pública o alguna zona restringida, los costos de tender cables privados por lo general son prohibitivos. Además, en casi todos los países del mundo también es ilegal el enlace de líneas de transmisión privadas a través (o por debajo) de una propiedad pública. En consecuencia, los diseñadores de redes deben depender de las instalaciones de telecomunicaciones existentes.

Por lo general, estas instalaciones, en especial la **PSTN** (**Red Telefónica Pública Conmutada),** fueron diseñadas hace muchos años, con un propósito completamente distinto: transmitir la voz humana en una forma más o menos reconocible. Su aplicabilidad en las comunicaciones de computadora a computadora es muy limitada, pero esta situación está cambiando rápidamente con la introducción de la fibra óptica y la tecnología digital. De cualquier manera, el sistema telefónico está tan estrechamente entrecruzado con las redes de computadoras (de área amplia) que bien vale la pena dedicarle un poco de tiempo a su estudio.

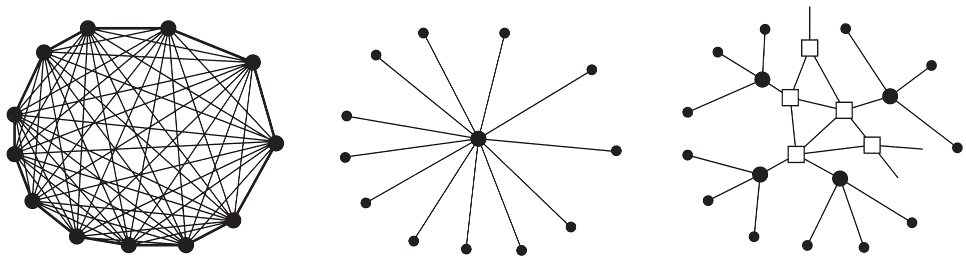
Con el propósito de entender la importancia del problema, realicemos una comparación burda pero ilustrativa de las propiedades de una conexión típica de computadora a computadora a través de un cable local y otra mediante una línea de acceso telefónico. Un cable entre dos computadoras puede transferir datos a 10^9 bps, o tal vez un poco más. En contraste, una línea de acceso telefónico tiene una tasa máxima de datos de 56 kbps, una diferencia de un factor de casi 20,000. Es como la diferencia entre un pato contoneándose campantemente entre la hierba y un cohete a la Luna. Si la línea de acceso telefónico se reemplaza por una conexión ADSL, sigue habiendo una diferencia de un factor de 1000-2000.

Por supuesto, el problema es que los diseñadores de sistemas de cómputo suelen trabajar con sistemas de cómputo y cuando de repente se enfrentan con un sistema cuyo desempeño (según lo que ellos piensan) es tres o cuatro órdenes de magnitud peor, ellos, lo cual no es una sorpresa,

dedican mucho tiempo y esfuerzo para tratar de averiguar cómo utilizarlo de manera eficiente. En las siguientes secciones describiremos el sistema telefónico y mostraremos cómo funciona. Para obtener mayor información sobre los aspectos técnicos del sistema telefónico vea (Bellamy, 2000).

2.5.1 Estructura del sistema telefónico

Tan pronto como Alexander Graham Bell patentó el teléfono en 1876 (tan sólo unas cuantas horas antes que su rival, Elisha Gray), hubo una gran demanda por su nuevo invento. El mercado inicial fue para la venta de teléfonos, los cuales se vendían en pares. Le tocaba al cliente conectarlos con un solo alambre. Los electrones regresaban por tierra. Si el propietario de un teléfono deseaba comunicarse con otros *n* propietarios de teléfono, tenía que enlazar alambres individuales a todas las *n* casas. Después de un año, las ciudades se cubrieron de alambres que pasaban sobre las casas y los árboles convirtiéndose en una maraña. De inmediato quedó en claro que el modelo de conexión de cada teléfono con todos los demás, como se muestra en la figura 2-20(a), no iba a funcionar.



(a) (b) (c)

Figura 2-20. (a) Red totalmente interconectada. (b) Conmutador centralizado. (c) Jerarquía de dos niveles.

Bell tuvo la suficiente visión para darse cuenta de esto y formó la Bell Telephone Company, la cual abrió su primera oficina de conmutación en 1878 (en New Haven, Connecticut). La compañía colocó un alambre en la casa u oficina de cada cliente. Para realizar una llamada, el cliente debía dar vueltas a una manivela en el teléfono para producir un sonido en la oficina de la compañía de teléfonos que atrajera la atención del operador, que a continuación conectaba manualmente a quien llamaba con el receptor de la llamada por medio de un cable puenteador. El modelo de la oficina de conmutación se muestra en la figura 2-20(b).

Muy pronto surgieron por todas partes oficinas de conmutación del Bell System y la gente quiso hacer llamadas de larga distancia entre ciudades, de modo que el Bell System empezó a conectar las oficinas de conmutación. El problema original pronto reapareció: conectar cada oficina de conmutación con todas las demás por medio de un cable entre ellas pronto dejó de ser práctico, así que se inventaron las oficinas de conmutación de segundo nivel. Poco después, fueron necesarias múltiples oficinas de segundo nivel, como se muestra en el diagrama de la figura 2-20(c). Por último, la jerarquía creció a cinco niveles.

Para 1890, las tres partes principales del sistema telefónico ya estaban en su lugar: las oficinas de conmutación, los cables entre los clientes y las oficinas de conmutación (a estas alturas cables de par trenzado balanceados y aislados, en lugar de cables abiertos con retorno a tierra) y las conexiones de larga distancia entre las oficinas de conmutación. Aunque desde entonces se han realizado mejoras en las tres áreas, el modelo básico del Bell System ha permanecido intacto en lo esencial por más de 100 años. Para una historia técnica corta del sistema telefónico vea (Hawley, 1991).

Previo a la división de AT&T en 1984, el sistema telefónico fue organizado como una jerarquía de múltiples niveles, con alta redundancia. A pesar de su simplicidad, la siguiente descripción da una idea de la situación. Cada teléfono tiene dos alambres de cobre que van directamente a la **oficina central local** de la compañía telefónica. Por lo general, la distancia va de 1 a 10 km, y en las ciudades es más corta que en las áreas rurales. Tan sólo en Estados Unidos existen cerca de 22,000 oficinas centrales. En el ámbito de las comunicaciones, las conexiones de dos alambres entre el teléfono de cada suscriptor y la oficina central se conocen como **circuito local.** Si los circuitos locales de todo el mundo se extendieran de extremo a extremo, llegarían a la Luna y regresarían a la Tierra 1000 veces.

En cierto momento, el 80% del valor del capital de AT&T fue el cobre en los circuitos locales. En efecto, AT&T era entonces la más grande mina de cobre del mundo. Por fortuna, este hecho no era muy conocido en la comunidad inversionista. De haberse sabido, algún pirata corporativo podría haber comprado la AT&T, cancelado todo el servicio telefónico en Estados Unidos, extraído todos los cables y vendido el cableado a algún refinador de cobre para obtener una ganancia rápida.

Si un suscriptor conectado a una oficina central determinada llama a otro suscriptor conectado a la misma oficina central, el mecanismo de conmutación dentro de la oficina establece una conexión eléctrica directa entre los dos circuitos locales. Esta conexión permanece intacta mientras dura la llamada.

Si el teléfono al que se llama está conectado a otra oficina central, se tiene que usar un procedimiento diferente. Cada oficina central tiene varias líneas salientes a uno o más centros de conmutación cercanos, llamados **oficinas interurbanas** (o, si están dentro de la misma área local, **oficinas en tándem**). Estas líneas se llaman **troncales de conexión interurbanas.** Si sucede que tanto la oficina central de quien llama como la de quien es llamado tienen una troncal de conexión a la misma oficina interurbana (algo muy probable si no están muy alejadas), la conexión se puede establecer dentro de la oficina interurbana. En la figura 2-20(c) se muestra una red telefónica que consiste únicamente en teléfonos (los puntos pequeños), oficinas centrales (los puntos grandes) y oficinas interurbanas (los cuadrados).

Si el que llama y el que es llamado no tienen una oficina interurbana en común, la trayectoria se deberá establecer en un nivel más alto de la jerarquía. Hay oficinas primarias, seccionales y regionales que forman una red que conecta a las oficinas interurbanas. Las centrales interurbanas, primarias, seccionales y regionales se comunican entre sí mediante **troncales interurbanas** de gran ancho de banda. La cantidad de tipos diferentes de centros de conmutación y su topología varían de país a país dependiendo de su densidad telefónica (por ejemplo, ¿pueden dos oficinas

seccionales tener una conexión directa o deben pasar por una oficina regional?). La figura 2-21 muestra cómo se podría enrutar una conexión de media distancia.

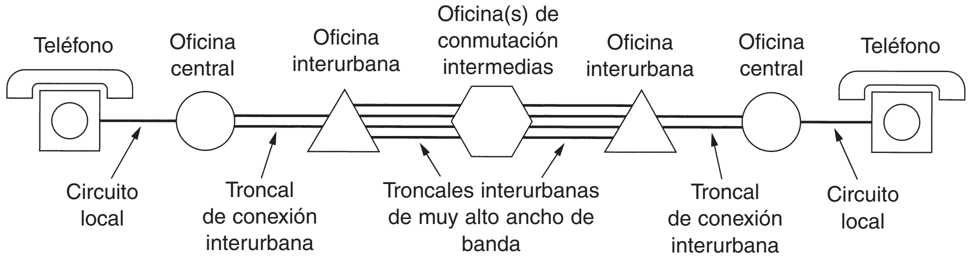


Figura 2-21. Ruta típica de un circuito para una llamada de media distancia.

Para telecomunicaciones se usan diversos medios de transmisión. En nuestros días, los circuitos locales consisten en pares trenzados, aunque en los primeros días de la telefonía eran comunes los cables no aislados espaciados a 25 cm en los postes telefónicos. Entre las oficinas de conmutación se usan ampliamente cables coaxiales, microondas y, en especial, fibra óptica.

En el pasado, la transmisión en todo el sistema telefónico era analógica, con la señal de voz real transmitida como un voltaje eléctrico entre la fuente y el destino. Con el advenimiento de la fibra óptica, la electrónica digital y las computadoras, actualmente todas las troncales y los conmutadores son digitales, y el circuito local queda como el único elemento de tecnología analógica del sistema. Existe preferencia por la transmisión digital porque en ésta no es necesario reproducir exactamente una forma de onda analógica después de que ha pasado por muchos amplificadores en una llamada larga. Es suficiente con distinguir correctamente un 0 de un 1. Esta propiedad da más confiabilidad a la transmisión digital que a la analógica. Su mantenimiento también es más económico y sencillo.

En síntesis, el sistema telefónico consiste en tres componentes principales:

1. Circuitos locales (cables de par trenzado que van hacia las casas y las empresas).
2. Troncales (fibra óptica digital que conecta a las oficinas de conmutación).
3. Oficinas de conmutación (donde las llamadas pasan de una troncal a otra).

Después de una breve digresión sobre la política de los teléfonos, regresaremos a cada uno de estos tres componentes en detalle. Los circuitos locales dan acceso a todo mundo al sistema completo, debido a lo cual son cruciales. Por desgracia, también son la parte más débil del sistema. Para las troncales de largo alcance, la principal consideración es cómo reunir múltiples llamadas y enviarlas juntas por la misma fibra. Este tema se llama multiplexión, y estudiaremos tres formas diferentes de hacerlo. Por último, existen dos formas fundamentalmente distintas de efectuar la conmutación, así que veremos ambas.

2.5.2 La política de los teléfonos

Durante las décadas anteriores a 1984, el Bell System proporcionó tanto el servicio local como el de larga distancia en casi todo Estado Unidos. En la década de 1970, el gobierno estadounidense se convenció de que éste era un monopolio ilegal y entabló un juicio para dividirlo. El gobierno ganó, y el 1o. de enero de 1984 la AT&T se dividió en AT&T Long Lines, 23 **BOCs** (**Compañías Operativas de Bell**) y algunas otras partes pequeñas. Las 23 BOCs se agruparon en siete BOCs regionales (RBOCs) para hacerlas económicamente viables. La naturaleza entera de la telecomunicación en Estados Unidos se cambió de la noche a la mañana por orden judicial (*no* por una ley del Congreso).

Los detalles exactos del desmantelamiento se describieron en el llamado **MFJ** (**Juicio Final Modificado**), un claro contrasentido (si el juicio se pudo modificar, obviamente no era final). Este suceso condujo a un aumento en la competencia, mejor servicio y menores precios en larga distancia para los consumidores y las empresas. No obstante, los precios del servicio local se elevaron cuando los subsidios a las llamadas de larga distancia se eliminaron y el servicio local tuvo que ser autosuficiente. Muchos otros países consideran ahora la introducción de la competencia por caminos similares.

Para dejar en claro quiénes podrían actuar y cómo, Estados Unidos se dividió en más de 160 **LATAs** (**Áreas de Acceso y Transporte Local**). En forma muy aproximada, una LATA es casi tan grande como el área cubierta por un código de área. Dentro de una LATA normalmente había una **LEC** (**Portadora de Intercambio Local**) que tenía un monopolio sobre el servicio tradicional de telefonía dentro de la LATA. Las LECs más importantes eran las BOCs, aunque algunas LATAs contenían una o más de las 1500 compañías telefónicas independientes que operaban como LECs.

Un tipo de compañía diferente maneja todo el tráfico interLATA: una **IXC** (**Portadora Entre Centrales**). Originalmente, AT&T Long Lines era la única IXC seria, pero ahora WorldCom y Sprint son competidores bien establecidos en el negocio de las IXCs. Una de las consideraciones durante la división fue asegurar que todas las IXCs serían tratadas con igualdad en términos de calidad de líneas, tarifas y cantidad de dígitos que tendrían que marcar sus clientes para usarlos. La forma como esto se resolvió se ilustra en la figura 2-22. Aquí vemos tres LATAs de ejemplo, cada una con varias oficinas centrales. Las LATAs 2 y 3 tienen también una pequeña jerarquía con oficinas en tándem (oficinas interurbanas intraLATA).

Cualquier IXC que desee manejar llamadas que se originen en una LATA puede construir allí una oficina de conmutación llamada **POP** (**Punto de Presencia**). La LEC es necesaria para conectar cada IXC a cada oficina central, ya sea en forma directa, como en las LATAs 1 y 3, o indirecta, como en la LATA 2. Además, los términos de la conexión, tanto técnicos como financieros, deben ser idénticos para todas las IXCs. De esta forma, un suscriptor en, digamos, la LATA 1 puede elegir cuál IXC usar para llamar a los suscriptores en la LATA 3.

Como parte del MFJ, se prohibió a las IXCs ofrecer servicio telefónico local y a las LECs ofrecer servicio telefónico interLATA, aunque ambas eran libres de participar en otros negocios, como la operación de restaurantes de pollos fritos. En 1984, éste era un dictamen bastante claro. Desgraciadamente, la tecnología tiene la mala costumbre de hacer obsoletas las leyes. Ni la televisión por cable ni los teléfonos celulares estaban considerados en el acuerdo. Conforme la te-

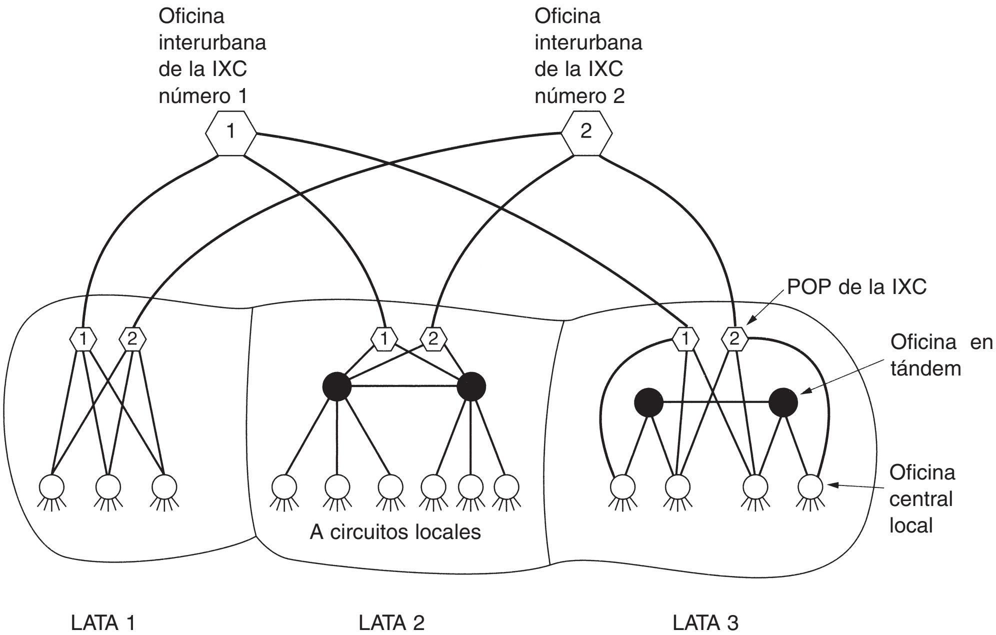


Figura 2-22. Relación entre LATAs, LECs e IXCs. Todos los círculos son oficinas de conmutación LEC. Cada hexágono pertenece a la IXC cuyo número contiene.

levisión por cable pasó de ser unidireccional a bidireccional, y la popularidad de los teléfonos celulares subió como la espuma, tanto las LECs como las IXCs comenzaron a comprar o a fusionarse con los operadores de cable y celulares.

Para 1995, el Congreso vio que tratar de mantener la distinción entre las diversas clases de compañías ya no era sostenible y esbozó una propuesta de ley para permitir a las compañías de televisión por cable, a las compañías telefónicas locales, a los operadores de larga distancia y a los operadores de teléfonos celulares participar en los negocios de unos y otros. La intención era que así cualquier compañía podría ofrecer a sus clientes un solo paquete integrado que contuviera televisión por cable, teléfono y servicios de información, y que las diferentes compañías compitieran en servicio y precio. La propuesta se convirtió en ley en febrero de 1996. Como resultado, algunas BOCs se convirtieron en IXCs y algunas otras compañías, como los operadores de televisión por cable, empezaron a competir con las LECs por el servicio telefónico local.

Un aspecto interesante de la ley de 1996 fue la obligación para las LECs de implementar portabilidad para los números locales. Esto quiere decir que un cliente puede cambiar de compañía telefónica local sin necesidad de obtener un nuevo número telefónico. Esta cláusula elimina un enorme obstáculo para los usuarios y los anima a cambiar de LEC, con lo cual se incrementa la competencia. En consecuencia, el panorama de las telecomunicaciones en Estados Unidos está atravesando una reestructuración radical. De nueva cuenta, muchos otros países están siguiendo esta línea. Con frencuencia, otros países esperan para ver cómo funciona esta clase de experimentos en Estados Unidos. Si da resultado, adoptan el esquema; si falla, buscan otras alternativas.

2.5.3 El circuito local: módems, ADSL e inalámbrico

Es hora de iniciar el estudio detallado del funcionamiento del sistema telefónico. Las principales partes del sistema se ilustran en la figura 2-23. Aquí vemos los circuitos locales, las troncales y las oficinas interurbanas y oficinas centrales, las cuales tienen equipo que conmuta las llamadas. Una oficina central tiene hasta 10,000 circuitos locales (en Estados Unidos y otros países grandes). De hecho, hasta hace poco tiempo el código de área + caracteres de sustitución indicaban la oficina central, de tal manera que (212) 601-xxxx se refería a una oficina central específica con 10,000 suscriptores, numerados de 0000 a 9999. Con el surgimiento de la competencia por el servicio local, este sistema dejó de ser funcional porque diversas compañías querían apoderarse del código de oficina central. Asimismo, el número de códigos prácticamente se había consumido, por lo que hubo necesidad de introducir esquemas de correspondencia complejos.

Empecemos con el tema que la mayoría de la gente conoce: el circuito local de dos alambres que parte de la oficina central de una compañía telefónica hacia hogares y pequeñas empresas. El circuito local se conoce también como de "última milla" (la conexión hacia el cliente), aunque la longitud puede ser de varias millas. Durante más de 100 años ha utilizado señalización analógica y es probable que continúe haciéndolo durante algún tiempo, debido al costo elevado de la conversión a digital. No obstante, el cambio se está dando incluso en este último bastión de la transmisión analógica. En esta sección estudiaremos el circuito local tradicional y los avances que están teniendo lugar, con especial atención en la comunicación de datos desde computadoras caseras.

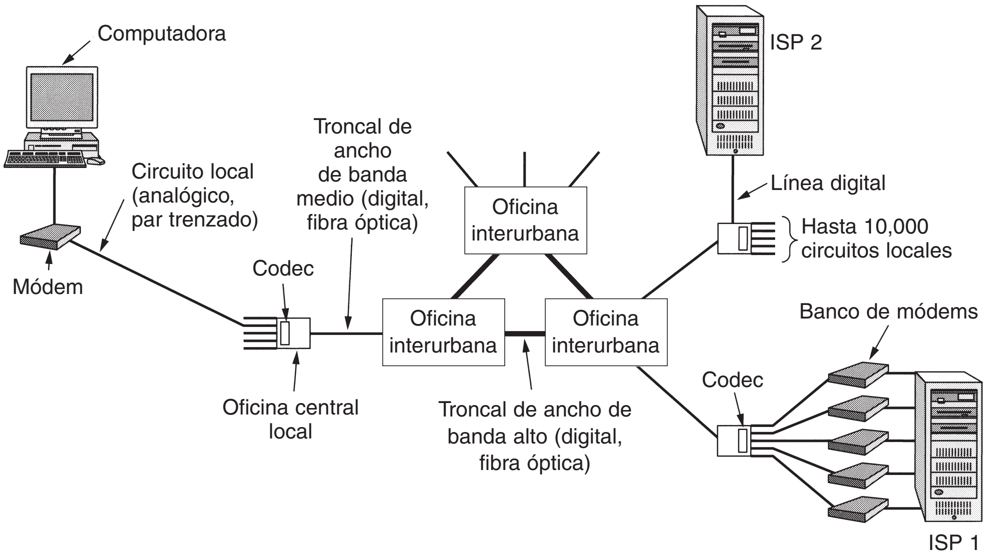


Figura 2-23. Uso de transmisión analógica y digital para una llamada de computadora a computadora. Los módems y los codecs realizan la conversión.

Cuando una computadora desea enviar datos digitales sobre una línea analógica de acceso telefónico, es necesario convertir primero los datos a formato analógico para transmitirlos sobre el

circuito local. Un dispositivo conocido como módem realiza esta conversión, tema que estudiaremos en breve. Los datos se convierten a formato digital en la oficina central de la compañía telefónica para transmitirlos sobre las troncales que abarcan largas distancias.

Si en el otro extremo hay una computadora con un módem, es necesario realizar la conversión inversa —digital a analógico— para recorrer el circuito local en el destino. Esta disposición se muestra en la figura 2-23 para el ISP 1 (proveedor de servicios de Internet), que tiene un banco de módems, cada uno conectado a un circuito local diferente. Este ISP puede manejar tantas conexiones como módems tenga (suponiendo que su servidor o sus servidores tengan suficiente potencia de cómputo). Esta disposición fue la común hasta que aparecieron los módems de 56 kbps, por razones que veremos más adelante.

La señalización analógica consiste en la variación del voltaje con el tiempo para representar un flujo de información. Si los medios de transmisión fueran perfectos, el receptor recibiría exactamente la misma señal enviada por el transmisor. Por desgracia, los medios no son perfectos, por lo cual la señal recibida no es la misma que la transmitida. Si los datos son digitales, esta diferencia puede conducir a errores.

Las líneas de transmisión tienen tres problemas principales: atenuación, distorsión por retardo y ruido. La **atenuación** es la pérdida de energía conforme la señal se propaga hacia su destino. La pérdida se expresa en decibeles por kilómetro. La cantidad de energía perdida depende de la frecuencia. Para ver el efecto de esta dependencia de la frecuencia, imagine una señal no como una simple forma de onda, sino como una serie de componentes de Fourier. Cada componente es atenuado en diferente medida, lo que da por resultado un espectro de Fourier distinto en el receptor.

Por si esto no fuera poco, los diferentes componentes de Fourier se propagan a diferente velocidad por el cable. Esta diferencia de velocidad ocasiona una **distorsión** de la señal que se recibe en el otro extremo.

Otro problema es el **ruido,** que es energía no deseada de fuentes distintas al transmisor. El movimiento al azar de los electrones en un cable causa el ruido térmico y es inevitable. La diafonía se debe al acoplamiento inductivo entre dos cables que están cerca uno de otro. A veces, al hablar por teléfono se escucha otra conversación en el fondo. Ésa es la diafonía. Finalmente, hay ruido de impulso, causado por picos en la línea de suministro de energía o por otros fenómenos. En el caso de datos digitales, el ruido de impulso puede eliminar uno o más bits.

Módems

Debido a los problemas antes mencionados, en especial al hecho de que tanto la atenuación como la velocidad de propagación dependen de la frecuencia, es indeseable tener un rango amplio de frecuencias en la señal. Desgraciadamente, las ondas cuadradas, como las de los datos digitales, tienen un espectro amplio y por ello están sujetas a una fuerte atenuación y a distorsión por retardo. Estos efectos hacen que la señalización de banda base (CC, corriente continua) sea inadecuada, excepto a velocidades bajas y distancias cortas.

La señalización de CA (corriente alterna) se utiliza para superar los problemas asociados a la señalización de CC, en especial en las líneas telefónicas. Se introduce un tono continuo en el rango de 1000 a 2000 Hz, llamado **portadora de onda senoidal,** cuya amplitud, frecuencia o fase se

pueden modular para transmitir la información. En la **modulación de amplitud** se usan dos niveles diferentes de amplitud para representar 0 y 1, respectivamente. En la **modulación de frecuencia,** conocida también como **modulación por desplazamiento de frecuencia,** se usan dos (o más) tonos diferentes. En la forma más simple de la **modulación de fase** la onda portadora se desplaza de modo sistemático 0 o 180 grados a intervalos espaciados de manera uniforme. Un mejor esquema es utilizar desplazamientos de 45, 135, 225 o 315 grados para transmitir 2 bits de información por intervalo. Asimismo, al requerir siempre un desplazamiento de fase al final de cada intervalo se facilita que el receptor reconozca los límites de los intervalos.

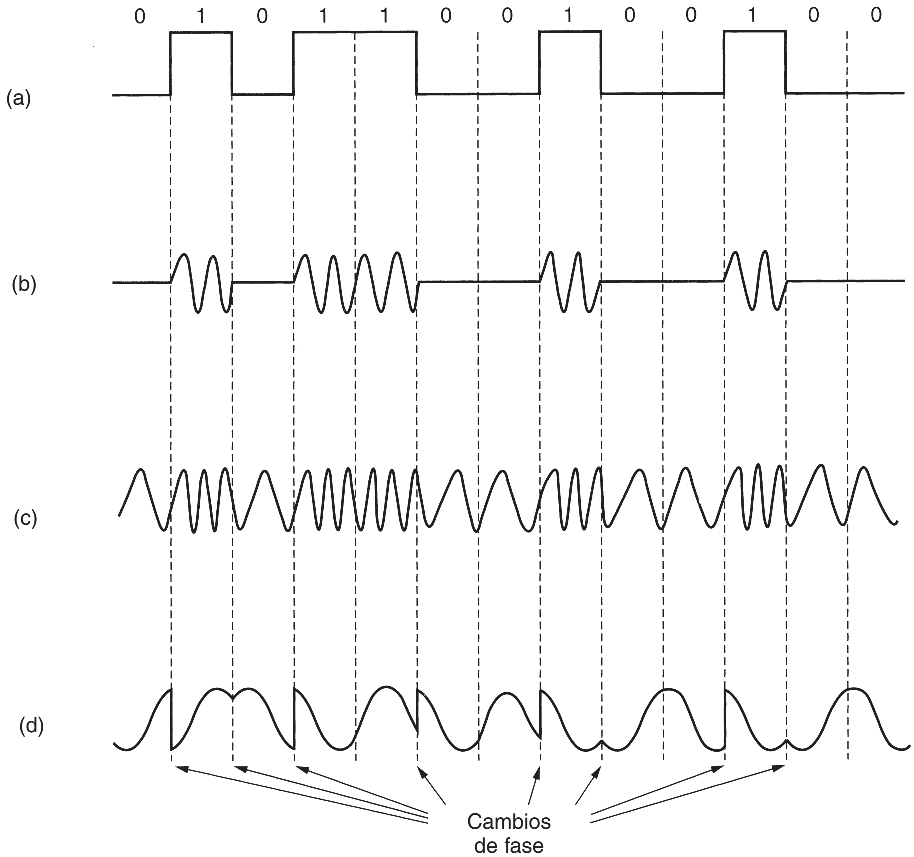


Figura 2-24. (a) Señal binaria. (b) Modulación de amplitud. (c) Modulación de frecuencia. (d) Modulación de fase.

La figura 2-24 ilustra los tres tipos de modulación. En la figura 2-24(a) una de las amplitudes es distinta de cero y la otra es cero. En la figura 2-24(b) se utilizan dos frecuencias. En la figura 2-24(c) está presente o ausente un desplazamiento de fase en cada límite de bit. Un **módem** (por modulador-demodulador) es un dispositivo que acepta un flujo de bits en serie como entrada y que

produce una portadora modulada mediante uno (o más) de estos métodos (o viceversa). El módem se conecta entre la computadora (digital) y el sistema telefónico (analógico).

Para alcanzar velocidades cada vez más altas, no basta sólo incrementar la velocidad de muestreo. El teorema de Nyquist dice que aun con una línea perfecta de 3000 Hz (cosa que decididamente no son las líneas de acceso telefónico), no tiene sentido muestrear más allá de 6000 Hz. En la práctica, la mayoría de los módems muestrea 2400 veces por segundo y el objetivo es conseguir más bits por muestra.

El número de muestras por segundo se mide en **baudios.** Un **símbolo** se envía durante cada baudio. De esta manera, una línea de *n* baudios transmite *n* símbolos por segundo. Por ejemplo, una línea de 2400 baudios envía un símbolo más o menos cada 416.667 µseg. Si el símbolo consta de 0 voltios para un 0 lógico y de 1 voltio para un 1 lógico, la tasa de bits es de 2400 bps. Sin embargo, si se utilizan los voltajes 0, 1, 2 y 3, cada símbolo consta de 2 bits, por lo que una línea de 2400 baudios pueden transmitir 2400 símbolos por segundo a una tasa de datos de 4800 bps. De manera similar, con cuatro posibles desplazamientos de fase también hay 2 bits por símbolo, con lo cual la tasa de bits es otra vez el doble que la tasa de baudios. La última técnica se utiliza ampliamente y se denomina **QPSK** (**Codificación por Desplazamiento de Fase en Cuadratura**).

Es común la confusión de los conceptos ancho de banda, baudio, símbolo y tasa de bits, por lo que los definiremos a continuación. El ancho de banda de un medio es el rango de frecuencias que atraviesa al medio con atenuación mínima. Es una propiedad física del medio (por lo general, de 0 a alguna frecuencia máxima) y se mide en hertzios (Hz). La tasa de baudios es la cantidad de muestras por segundo que se realizan. Cada muestra envía una porción de información, es decir, un símbolo. Por lo tanto, la tasa de baudios y la tasa de símbolos significan lo mismo. La técnica de modulación (por ejemplo, QPSK) determina la cantidad de bits por símbolo. La tasa de bits es la cantidad de información que se envía por el canal y es igual a la cantidad de símbolos por segundo por la cantidad de bits por símbolo.

Todos los módems avanzados utilizan una combinación de técnicas de modulación con el propósito de transmitir muchos bits por baudio. Con frecuencia se combinan múltiples amplitudes y múltiples desplazamientos de fase para transmitir muchos bits por símbolo. En la figura 2-25(a) vemos puntos con amplitud constante a los 45, 135, 225 y 315 grados (distancia desde el origen). La fase de un punto la indica el ángulo que se forma con el eje de las X al trazar una línea desde el punto hacia el origen. La figura 2-25(a) tiene cuatro combinaciones válidas y se puede utilizar para transmitir 2 bits por símbolo. Es QPSK.

En la figura 2-25(b) se muestra un esquema de modulación distinto, en el cual se utilizan cuatro amplitudes y cuatro fases, que permiten un total de 16 combinaciones diferentes. Este esquema de modulación se puede utilizar para transmitir 4 bits por símbolo. Se conoce como **QAM-16** (**Modulación de Amplitud en Cuadratura**). En algunas ocasiones también se utiliza el término **16-QAM.** Por ejemplo, QAM-16 se puede utilizar para transmitir 9600 bps sobre una línea de 2400 baudios.

En la figura 2-25(c) se presenta otro esquema de modulación que incluye amplitud y fase. En éste se pueden conseguir 64 combinaciones diferentes, por lo cual es posible transmitir 6 bits por símbolo. Se conoce como **QAM-64.** También se utilizan QAMs de orden más alto.

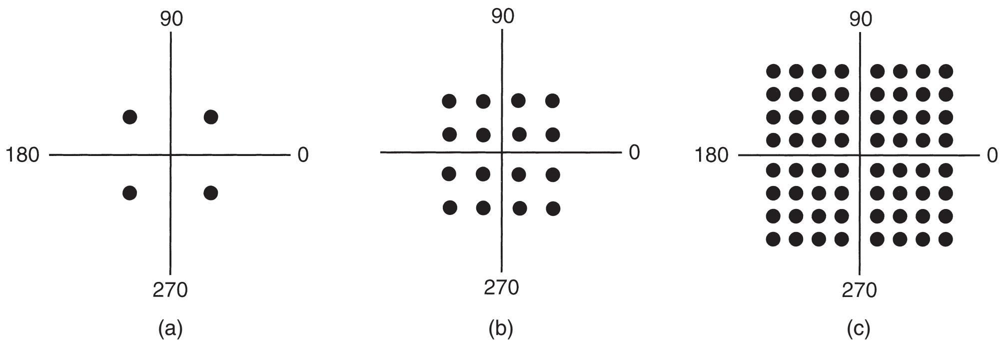


Figura 2-25. (a) QPSK. (b) QAM-16. (c) QAM-64.

A los diagramas como los de la figura 2-25, que muestran las combinaciones permitidas de amplitud y fase, se les llama **diagramas de constelación.** Cada estándar de módem de alta velocidad tiene su propio diagrama de constelación y se puede comunicar solamente con otros módems que utilicen el mismo modelo (aunque la mayoría de los módems puede emular a todos los modelos más lentos).

Cuando hay muchos puntos en un diagrama de constelación, incluso la cantidad mínima de ruido en la amplitud o fase detectada puede dar como resultado un error y, potencialmente, muchos bits malos. Con el propósito de reducir la posibilidad de error, los estándares para los módems de velocidades más altas realizan corrección de errores mediante la incorporación de bits adicionales en cada muestra. Los esquemas se conocen como **TCM** (**Modulación por Codificación de Malla).** Así, por ejemplo, el estándar V.32 de módem utiliza 32 puntos de constelación para transmitir 4 bits de datos y 1 bit de paridad por símbolo a 2400 baudios, para alcanzar 9600 bps con corrección de errores. Su diagrama de constelación se muestra en la figura 2-26(a). La decisión de "girar" 45 grados alrededor del origen se tomó por razones de ingeniería; las constelaciones giradas y sin girar tienen la misma capacidad de información.

El siguiente escalón después de 9600 bps es 14,400 bps. Se conoce como **V.32 bis.** Esta velocidad se alcanza al transmitir 6 bits de datos y 1 bit de paridad por muestra a 2400 baudios. Su diagrama de constelación tiene 128 puntos cuando se utiliza QAM-128, y se muestra en la figura 2-26(b). Los fax-módems transmiten a esta velocidad las páginas que han sido digitalizadas como mapas de bits. QAM-256 no se utiliza en ningún módem telefónico estándar, pero sí en redes de cable, como veremos más adelante.

Enseguida del módem telefónico V.32 se encuentra el **V.34,** el cual corre a 28,800 bps, 2400 baudios y 12 bits de datos por símbolo. El último módem de esta serie es el **V.34 bis,** el cual transfiere 14 bits de datos por símbolo a 2400 baudios para alcanzar una velocidad de 33,600 bps.

Para incrementar aún más la tasa de datos efectiva, muchos módems comprimen los datos antes de enviarlos, y alcanzan tasas de datos efectivas mayores a 33,600 bps. Por otra parte, casi todos los módems prueban la línea antes de empezar a transmitir datos del usuario, y si encuentran una falta de calidad, reducen la velocidad a una menor a la máxima que tiene asignada. Por lo tanto, la velocidad *efectiva* del módem que percibe el usuario puede ser menor, igual o mayor a la que oficialmente tiene asignada.

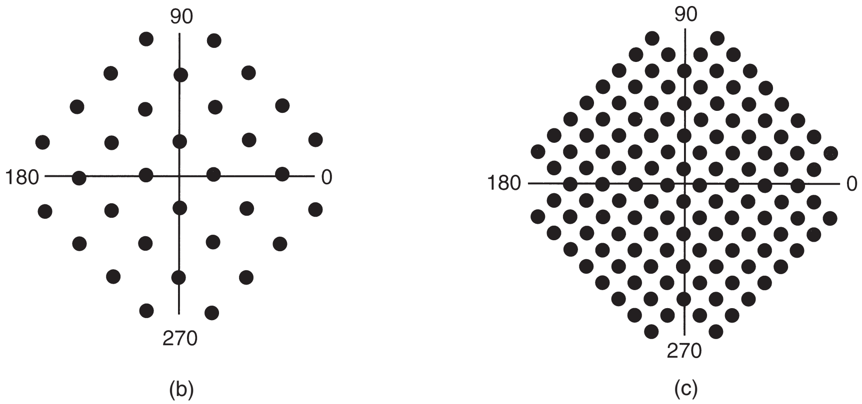


Figura 2-26. (a) V.32 para 9600 bps. (b) V32 bis para 14,400 bps.

Todos los módems modernos transmiten tráfico en ambas direcciones al mismo tiempo (mediante el uso de frecuencias distintas para las diferentes direcciones). La conexión que permite el flujo de tráfico en ambas direcciones de manera simultánea se conoce como **dúplex total.** Una carretera de dos carriles es dúplex total. La conexión que permite el tráfico en ambas direcciones, pero sólo en un sentido a la vez, se denomina **semidúplex.** Una vía de ferrocarril es semidúplex. La conexión que permite el tráfico en una sola dirección se conoce como **símplex.** Una calle de un solo sentido es símplex. Otro ejemplo de una conexión símplex lo constituye una fibra óptica con un láser en un extremo y un detector de luz en el otro.

La razón por la cual los módems estándar llegan hasta 33,600 es que el límite de Shannon para el sistema telefónico es de aproximadamente 35 kbps, así que velocidades mayores a este límite violarían las leyes de la física (departamento de termodinámica). Para saber si los módems de 56 kbps son posibles desde un punto de vista teórico, continúe leyendo.

¿Pero a qué se debe que el límite teórico sea de 35 kbps? La respuesta está en la longitud promedio de los circuitos locales y en la calidad de estas líneas. La longitud promedio de los circuitos locales determina los 35 kbps. En la figura 2-23, una llamada que se origina en la computadora de la izquierda y que termina en el ISP 1 recorre dos circuitos locales como señal analógica, una vez en el punto de origen y otra en el punto de destino. En cada uno de estos circuitos se agrega ruido a la señal. Si pudiéramos prescindir de uno de estos circuitos locales, podría duplicarse la tasa máxima.

El ISP 2 hace precisamente esto. Cuenta con una alimentación digital pura proveniente de la oficina central más cercana. La señal digital que se utiliza en las troncales es alimentada directamente al ISP 2, con lo cual se elimina la necesidad de codecs, módems y transmisión analógica en su extremo. Así, cuando un extremo de la conexión es puramente digital, como ocurre con la mayoría de los ISPs actuales, la tasa máxima de datos puede ser de hasta 70 kbps. El máximo entre dos usuarios caseros con líneas analógicas es de 33.6 kbps.

La razón por la cual se utilizan los módems de 56 kbps se relaciona con el teorema de Nyquist. El canal telefónico tiene un ancho de alrededor de 4000 Hz (incluyendo las bandas de protección o guarda). De esta forma, la cantidad máxima de muestras independientes por segundo

es de 8000. La cantidad de bits por muestra en Estados Unidos es de 8, uno de los cuales se utiliza con propósitos de control, con lo cual es posible transmitir 56,000 bits por segundo de datos de usuario. En Europa los 8 bits están disponibles para los usuarios, lo cual permitiría utilizar módems de 64,000 bits por segundo, pero se eligió la cifra de 56,000 para apegarse a un estándar internacional.

Este estándar para módems se denomina **V.90.** Hace posible un canal ascendente o de subida (del usuario al ISP) de 33.6 kpbs y un canal descendente o de bajada (del ISP al usuario) de 56 kbps, debido a que por lo regular hay más transporte de datos del ISP al usuario que al revés (por ejemplo, la solicitud de una página Web requiere sólo algunos bytes, pero el envío de la misma puede constituir varios megabytes). En teoría, podría ser factible un canal ascendente de más de 33.6 kbps de ancho, pero como muchos circuitos locales son demasiado ruidosos incluso para 33.6 kbps, se decidió asignar más ancho de banda al canal descendente para incrementar las posibilidades de que funcione en realidad a 56 kbps.

El paso siguiente al V.90 es el **V.92.** Estos módems tienen capacidad de 48 kbps en el canal ascendente si la línea puede manejarlo. También determinan la velocidad apropiada que se utilizará en alrededor de la mitad de los 30 segundos en que lo hacen los módems más antiguos. Por último, permiten que una llamada telefónica entrante interrumpa una sesión en Internet, siempre y cuando la línea tenga el servicio de llamada en espera.

Líneas digitales de suscriptor

Cuando la industria telefónica alcanzó por fin los 56 kbps, se congratuló a sí misma por haber realizado un buen logro. Mientras tanto, la industria de TV por cable ofrecía velocidades de hasta 10 Mbps en cables compartidos, y las compañías de satélite planeaban ofrecer más allá de 50 Mbps. Conforme el acceso a Internet se tornaba una parte importante de su negocio, las compañías telefónicas (LECs) se dieron cuenta de que necesitaban un producto más competitivo. En respuesta comenzaron a ofrecer nuevos servicios digitales sobre el circuito local. Los servicios con mayor ancho de banda que el servicio telefónico común se denominan en ocasiones como de **banda ancha,** aunque en realidad el término es más un concepto de marketing que un concepto técnico específico.

En un principio había muchas ofertas que se traslapaban, todas bajo el nombre general de **xDSL** (**Línea Digital de Suscriptor**)**,** por diversos *x.* Más adelante analizaremos estos servicios, pero primero nos enfocaremos en el que tal vez se convierta en el más popular: **ADSL** (**DSL Asimétrica**)**.** Debido a que ADSL aún está en desarrollo y no todos los estándares están totalmente establecidos, algunos de los detalles que mencionaremos podrían cambiar con el paso del tiempo, aunque el panorama general debe permanecer igual. Para obtener mayor información sobre ADSL, vea (Summers, 1999, y Vetter y cols., 2000).

La razón por la cual los módems son tan lentos es que los teléfonos fueron creados para transportar la voz humana y todo el sistema se ha optimizado cuidadosamente con este propósito. Los datos siempre han sido un aspecto secundario. En el lugar donde cada circuito local termina en la oficina central, el cable pasa a través de un filtro que atenúa todas las frecuencias abajo de 300 Hz y arriba de 3400 Hz. El corte no es abrupto —300 Hz y 3400 Hz son los puntos a 3 dB—, de tal

manera que el ancho de banda se indica como 4000 Hz aun cuando la distancia entre los puntos a 3 dB es de 3100 Hz. Por lo tanto, los datos también se restringen a esta banda estrecha.

El truco para que xDSL funcione es que cuando un cliente se suscribe al servicio, la línea de entrada se conecta a un tipo distinto de conmutador, que no cuenta con el filtro, gracias a lo cual toda la capacidad del circuito local queda disponible. En esta situación, el ancho de banda artificial de 3100 Hz generado por el filtro ya no es el factor limitante, sino el medio físico del circuito local.

Por desgracia, la capacidad del circuito local depende de varios factores, entre ellos su longitud, espesor y calidad general. En la figura 2-27 se muestra una gráfica del ancho de banda potencial como una función de la distancia. En esta figura se da por sentado que todos los demás factores son óptimos (cables nuevos, haces moderados de cables, etcétera).

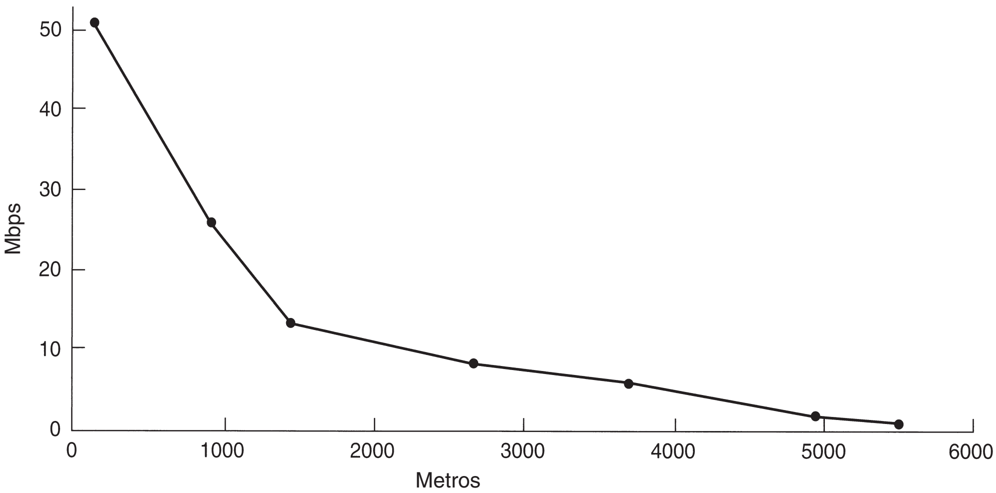


Figura 2-27. Ancho de banda contra distancia sobre la categoría 3 UTP para DSL.

La implicación de esta figura crea un problema para las compañías telefónicas. Cuando eligen la velocidad que ofrecerán, al mismo tiempo eligen un radio a partir de sus oficinas centrales más allá del cual no pueden proporcionar el servicio. Esto quiere decir que cuando un cliente distante intenta adquirir el servicio, podría obtener la siguiente respuesta: “Muchas gracias por su interés, pero no podemos darle el servicio porque usted vive 100 metros más lejos de la oficina central más cercana. ¿Podría mudarse?” Entre más baja sea la velocidad elegida, más amplio será el radio y podrán abarcarse más clientes. Pero entre más baja sea la velocidad, el servicio será menos atractivo y será menos la gente dispuesta a pagar por él. Aquí es donde se encuentran los negocios y la tecnología. (Una posible solución es construir minioficinas centrales en los vecindarios, pero es una alternativa costosa.)

Todos los servicios xDSL se diseñaron para que cumplieran algunos objetivos. Primero, los servicios deben funcionar sobre los circuitos locales existentes de par trenzado, categoría 3. Segundo, no deben afectar las máquinas de fax ni los teléfonos existentes de los clientes. Tercero, deben superar por mucho los 56 kbps. Cuarto, siempre deben funcionar, con sólo una tarifa mensual, no por minuto.

AT&T hizo la oferta inicial de ADSL, el cual funcionaba dividiendo el espectro disponible en el circuito local, de alrededor de 1.1 MHz, en tres bandas de frecuencia: **POTS** (**Servicio Telefónico Convencional**), canal ascendente (del usuario a la oficina central) y canal descendente (de la oficina central al usuario). La técnica en la cual se cuenta con múltiples bandas de frecuencia se conoce como multiplexión por división de frecuencia; en una sección posterior la analizaremos con detalle. Las ofertas subsecuentes de otros proveedores han tomado un enfoque distinto, y al parecer el siguiente es el probable ganador, así que lo describiremos a continuación.

El enfoque alternativo, llamado **DMT** (**MultiTono Discreto**), se ilustra en la figura 2-28. En efecto, lo que hace es dividir el espectro disponible de 1.1 MHz en el circuito local en 256 canales independientes de 4312.5 Hz cada uno. El canal 0 se utiliza para el POTS. Los canales 1-5 no se emplean, con el propósito de evitar que las señales de voz y de datos interfieran entre sí. De los 250 canales restantes, uno se utiliza para control del flujo ascendente y uno para control del flujo descendente. El resto está disponible para datos del usuario.

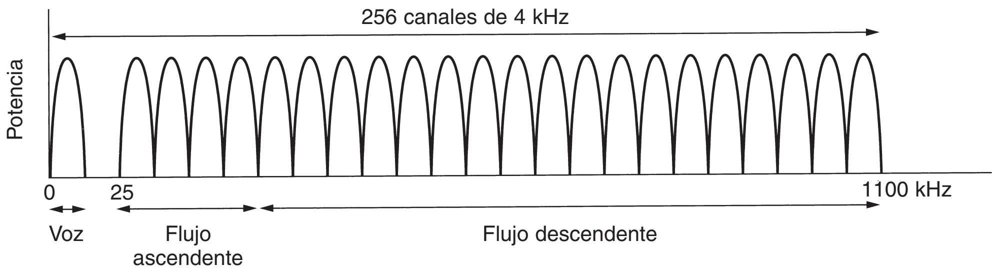


Figura 2-28. Operación de ADSL con modulación multitono discreta.

En principio, cada uno de los canales restantes se puede utilizar para un flujo de datos de dúplex total, pero las armónicas, la diafonía y otros efectos mantienen a los sistemas en la práctica muy por debajo del límite teórico. Queda a cargo del proveedor determinar cuántos canales se utilizarán para el flujo ascendente y cuántos para el flujo descendente. Es técnicamente posible una combinación de 50-50 de flujo ascendente y flujo descendente, pero la mayoría de los proveedores asigna entre 80 y 90% del ancho de banda al canal descendente debido a que el grueso de los usuarios descargan más datos que los que envían. Esta situación dio lugar a la "A" (asimétrica) de ADSL. Una división común es asignar 32 canales para el flujo ascendente y el resto al flujo descendente. También es posible establecer algunos de los canales de flujo ascendente más altos como bidireccionales para el ancho de banda incrementado, aunque esta optimización requiere agregar un circuito especial para cancelar el eco.

El estándar ADSL (ANSI T1.413 y el ITU G.992.1) permite velocidades de hasta 8 Mbps para el flujo descendente y de 1 Mbps para el flujo ascendente. No obstante, pocos proveedores ofrecen esta velocidad. Por lo general, los proveedores ofrecen 512 kbps para el flujo descendente y 64 kbps para el flujo ascendente (en el servicio estándar) y 1 Mbps para el flujo descendente y 256 kbps para el flujo ascendente (en el servicio premium).

Dentro de cada canal se utiliza un esquema de modulación similar a V.34, aunque la tasa de muestreo es de 4000 baudios en vez de 2400. La calidad de la línea en cada canal se monitorea

de manera constante y la tasa de datos se ajusta cada vez que es necesario, por lo cual canales distintos podrían tener tasas de datos diferentes. Los datos actuales se envían con modulación QAM, con un máximo de 15 bits por baudio, utilizando un diagrama de constelación análogo al de la figura 2-25(b). Por ejemplo, con 224 canales descendentes y 15 bits/baudio a 4000 baudios, el ancho de banda del flujo descendente es de 13.44 Mbps. En la práctica, la relación señal a ruido nunca es suficientemente buena para alcanzar esta tasa, pero en trayectorias cortas sobre circuitos de alta calidad es posible lograr 8 Mbps, razón por la cual el estándar llega hasta este punto.

En la figura 2-29 se muestra una disposición ADSL común. En este esquema, un técnico de la compañía telefónica debe instalar un **NID** (**Dispositivo de Interfaz de Red**) en la residencia del cliente. Esta pequeña caja de plástico delimita el fin de la propiedad de la compañía telefónica y el inicio de la propiedad del cliente. Cerca del NID (o en ocasiones en combinación con él) hay un **divisor,** un filtro analógico que separa la banda de 0-4000 Hz utilizada por la voz (POTS) de los datos. La señal POTS se enruta hacia el teléfono o máquina de fax existente, y la señal de datos se enruta a un módem. El módem ADSL es en realidad un procesador de señales digitales configurado para funcionar como 250 módems QAM operando en paralelo a diferentes frecuencias. Debido a que la mayoría de los módems ADSL actuales son externos, la computadora debe estar conectada a él a una velocidad alta. Por lo general, esto se consigue al colocar una tarjeta Ethernet en la computadora y poner a funcionar una Ethernet bastante corta de dos nodos tan sólo con la computadora y el módem ADSL. En ocasiones se utiliza el puerto USB en lugar de Ethernet. Sin duda, las tarjetas internas para módem ADSL estarán disponibles a futuro.

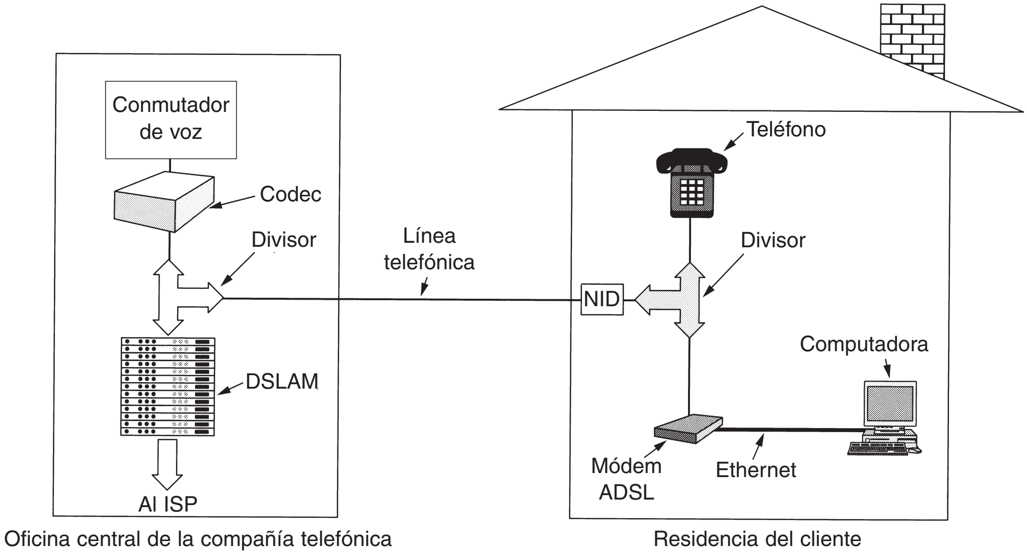


Figura 2-29. Configuración típica de un equipo ADSL.

En el otro extremo del cable, en la oficina central, se instala un divisor correspondiente. Aquí, se filtra la porción de voz de la señal y se envía al conmutador de voz normal. Las señales por arriba

de 26 kHZ se enrutan hacia un nuevo tipo de dispositivo conocido como **DSLAM** (**Multiplexor de Acceso de Línea Digital de Suscriptor**), el cual contiene el mismo tipo de procesador digital de señales que el módem ADSL. Una vez que la señal digital se extrae de un flujo de bits, se elaboran paquetes y se envían al ISP.

Esta completa separación entre el sistema de voz y ADSL facilita relativamente a la compañía telefónica el despliegue de ADSL. Todo lo que tiene que hacer es comprar un DSLAM y un divisor y conectar a los suscriptores ADSL al divisor. Otros servicios de ancho de banda alto (por ejemplo, ISDN) requieren cambios mucho más significativos al equipo de conmutación existente.

Una desventaja del diseño de la figura 2-29 es la presencia del NID y el divisor en la residencia del cliente. La instalación de estos componentes en la residencia del cliente sólo puede realizarla un técnico de la compañía telefónica, lo cual resulta bastante costoso. En consecuencia, también se ha estandarizado un diseño alternativo sin divisor. Informalmente se le conoce como G.lite, pero el número de estándar ITU es G.992.2. Es el mismo que el de la figura 2-29, aunque sin el divisor. La línea telefónica existente se utiliza tal como está. La única diferencia es que se tiene que colocar un microfiltro en cada conector telefónico, entre el teléfono o el módem ADSL y el cable. El microfiltro para el teléfono es un filtro pasa bajas que elimina frecuencias por arriba de 3400 Hz; el microfiltro para el módem ADSL es un filtro pasa altas que elimina las frecuencias por abajo de 26 kHz. El inconveniente es que este sistema no es tan confiable como el de divisor, por lo que G.lite sólo se puede utilizar hasta 1.5 Mbps (en comparación con los 8 Mbps para ADSL con un divisor). No obstante, G.lite aún requiere un divisor en la oficina central pero este tipo de instalación es relativamente económica y sencilla.

ADSL es tan sólo un estándar de la capa física. Lo que se ejecuta encima de él depende de la empresa portadora. Con frecuencia, ATM es la elección debido a su capacidad para manejar calidad de servicio y al hecho de que muchas compañías telefónicas ejecutan ATM en la red central.

Circuitos locales inalámbricos

Desde 1996 en Estados Unidos y un poco más tarde en otros países, existe libertad para las compañías que desean entrar a la competencia con la compañía telefónica local (antes monopolista), llamada **ILEC** (**LEC Obligada**). Las candidatas más probables son las compañías telefónicas de larga distancia (IXCs). Cualquier IXC que desee entrar al negocio telefónico local en alguna ciudad debe hacer lo siguiente: primero, debe comprar o alquilar un edificio para establecer su primera oficina central en dicha ciudad. Segundo, debe equipar la oficina con conmutadores telefónicos y otros dispositivos, todos los cuales están disponibles para venta directa al público. Tercero, debe tender una conexión de fibra óptica entre la oficina central y su central interurbana más cercana para que los clientes locales tengan acceso a su red nacional. Cuarto, debe conseguir clientes, por lo general, promoviendo un mejor servicio o precios más bajos que los de la ILEC.

Aquí empieza la parte difícil. Suponga que la compañía consigue algunos clientes. ¿De qué manera la nueva compañía telefónica local, conocida como **CLEC** (**LEC Competitiva**), conectará los teléfonos y computadoras de los clientes a su flamante nueva oficina central? La adquisición de los derechos de paso necesarios y el tendido de los cables o fibras son extremadamente costosos.

Muchas CLECs han encontrado una alternativa de bajo costo en lugar del tradicional circuito local con cable de par trenzado: el **WLL** (**Circuito Local Inalámbrico**).

De cierta manera, un teléfono fijo que utiliza un circuito local inalámbrico se parece un poco a un teléfono móvil, pero existen tres diferencias técnicas importantes. Primera, el cliente del circuito local inalámbrico con frecuencia desea conectividad de alta velocidad a Internet, al menos similar a la de ADSL. Segunda, al nuevo cliente probablemente no le importe que un técnico de la CLEC tenga que instalar una gran antena direccional en su techo, la cual apunta a la oficina central de la CLEC. Tercera, el usuario no se mueve, con lo cual se evitan todos los problemas asociados a la movilidad y la transferencia de celdas (*cell handoff*) que estudiaremos más tarde en este capítulo. Por lo tanto, estamos ante el surgimiento de una nueva industria: la **inalámbrica fija** (teléfono local y servicio de Internet ofrecidos por CLECs sobre circuitos locales inalámbricos).

Aunque los WLLs empezaron a funcionar de manera formal en 1998, debemos remontarnos a 1969 para conocer su origen. En ese año la FCC asignó dos canales de televisión (a 6 MHz cada uno) para la televisión educativa a 2.1 GHz. En años posteriores se agregaron 31 canales más a 2.5 GHz para totalizar 198 MHz.

La televisión educativa nunca se popularizó y en 1998 la FCC decidió quitarle las frecuencias y asignarlas a la radio bidireccional. De inmediato fueron utilizadas para los circuitos locales inalámbricos. A estas frecuencias, las microondas tienen una longitud de 10-12 cm. Poseen un rango de casi 50 km y pueden penetrar la vegetación y la lluvia moderadamente bien. Los 198 MHz de nuevo espectro fueron puestos inmediatamente en uso para los circuitos locales inalámbricos en un servicio denominado **MMDS** (**Servicio de Distribución Multipunto y Multicanal**). El MMDS se puede considerar como una MAN (red de área metropolitana), al igual que su similar LMDS (que se analiza más abajo).

La gran ventaja de este servicio es que la tecnología está bien desarrollada y que el equipo se consigue con facilidad. La desventaja consiste en que el ancho de banda total disponible es modesto y deben compartirlo muchos usuarios de una enorme área geográfica.

El bajo ancho de banda del MMDS despertó el interés en las ondas milimétricas como alternativa. No se asignaron frecuencias en el rango de 28-31 GHz en Estados Unidos y de 40 GHz en Europa debido a la dificultad de construir circuitos integrados de silicio que operen a esas velocidades. El problema fue resuelto con la invención de circuitos integrados de arseniuro de galio, lo que abrió las bandas milimétricas para la radiocomunicación. La FCC respondió a la demanda al asignar 1.3 GHz a un nuevo servicio de circuito local inalámbrico llamado **LMDS** (**Servicio Local de Distribución Multipunto**). Esta porción de ancho de banda es la mayor que la FCC ha asignado de una sola vez para cualquier uso. En Europa se asignó una porción similar, aunque a 40 GHz.

En la figura 2-30 se muestra cómo funciona LMDS. Se puede apreciar una torre con varias antenas, cada una de las cuales apunta a una dirección distinta. Debido a que las ondas milimétricas son altamente direccionales, cada antena define un sector, independiente de los demás. A esta frecuencia, el rango es de 2-5 km, lo cual quiere decir que se necesitan muchas antenas para abarcar una ciudad.

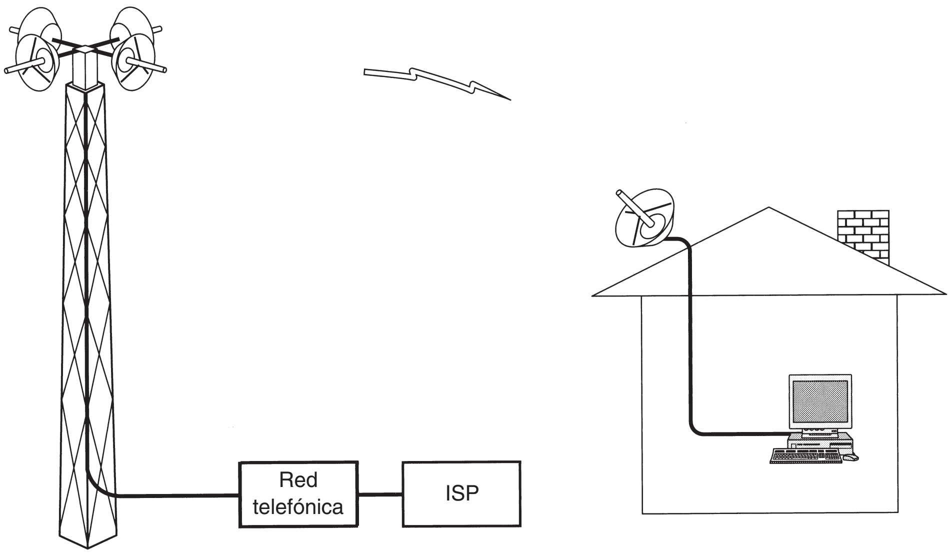


Figura 2-30. Arquitectura de un sistema LMDS.

Al igual que ADSL, LMDS asigna el ancho de banda de manera asimétrica, dando prioridad al canal descendente. Con la tecnología actual, cada sector puede contar con 36 Gbps de flujo descendente y 1 Mbsp de flujo ascendente, compartidos por todos los usuarios del sector. Si cada usuario activo descarga 3 páginas de 5 KB por minuto, el usuario ocupa un promedio de 2000 bps de espectro, lo cual permite un máximo de 18,000 usuarios activos por sector. No obstante, para mantener un retardo razonable debe haber un máximo de 9000 usuarios activos. Con cuatro sectores, como se muestra en la figura 2-30, puede soportarse una población de 36,000 usuarios activos. Suponiendo que uno de tres clientes esté en línea durante las horas de máxima actividad, una torre con cuatro antenas puede dar servicio a 100,000 usuarios dentro de un radio de 5 km de la torre. Muchas CLECs potenciales han realizado estos cálculos, y algunas de ellas han llegado a la conclusión de que con la cantidad necesaria para realizar una modesta inversión en torres de ondas milimétricas, se pueden meter al negocio de la telefonía local e Internet y ofrecer tasas de datos comparables a las de la televisión por cable, incluso a un menor precio.

Sin embargo, LMDS tiene algunos problemas. Por una parte, las ondas milimétricas se propagan en línea recta, por lo cual debe haber una línea visual despejada entre las antenas colocadas en el techo y la torre. Por otra parte, las hojas absorben bien estas ondas, por lo tanto, la torre debe tener suficiente altura para evitar los árboles en la línea visual. Lo que podría parecer una línea visual despejada en diciembre, tal vez no esté despejada en julio cuando los árboles están repletos de hojas. La lluvia también absorbe estas ondas. Hasta cierto punto, los errores producidos por la lluvia se pueden compensar con códigos de corrección de errores o incrementando la potencia cuando llueva. Con todo, es más probable que el servicio LMDS se estrene primero en climas secos, como en Arizona en vez de en Seattle.

Es poco probable que los circuitos locales inalámbricos se popularicen si no surgen estándares que animen a los fabricantes a producir equipo y que aseguren a los usuarios la oportunidad de cambiar de CLEC sin necesidad de comprar equipo nuevo. Con el propósito de proporcionar esta estandarización, el IEEE estableció el comité 802.16 para que se encargara de preparar el estándar para LMDS. El estándar 802.16 se publicó en abril de 2002. El IEEE denomina **MAN inalámbrica** al 802.16.

El estándar 802.16 del IEEE se diseñó para telefonía digital, acceso a Internet, conexión de dos LANs remotas, difusión por televisión y radio, entre otros usos. En el capítulo 4 lo veremos con más detalle.

2.5.4 Troncales y multiplexión

La economía de escala desempeña un papel importante en el sistema telefónico. Cuesta prácticamente lo mismo instalar y mantener una troncal de ancho de banda alto que una de ancho de banda bajo entre dos oficinas de conmutación (es decir, el gasto principal es la excavación de zanjas y no el cable de cobre o la fibra óptica). En consecuencia, las compañías telefónicas han desarrollado esquemas complejos para multiplexar muchas conversaciones en una sola troncal física. Estos esquemas de multiplexión se pueden dividir en dos categorías principales: **FDM** (**Multiplexión por División de Frecuencia**) y **TDM** (**Multiplexión por División de Tiempo**)**.** En FDM el espectro de frecuencia se divide en bandas de frecuencia, y cada usuario posee exclusivamente alguna banda. En TDM los usuarios esperan su turno (en *round-robin*), y cada uno obtiene en forma periódica toda la banda durante un breve lapso de tiempo.

La radiodifusión AM ilustra ambas clases de multiplexión. El espectro asignado es de alrededor de 1 MHz, aproximadamente de 500 a 1500 kHz. A los diferentes canales lógicos (estaciones) se les asigna una frecuencia distinta, y cada uno funciona en una porción del espectro con una separación entre canales lo bastante grande para evitar la interferencia. Este sistema es un ejemplo de multiplexión por división de frecuencia. Además (en algunos países), las estaciones individuales tienen dos subcanales lógicos: música y publicidad. Éstos se alternan en la misma frecuencia, primero una ráfaga de música y después una ráfaga de publicidad, luego más música, y así sucesivamente. Esta situación es multiplexión por división de tiempo.

A continuación examinaremos la multiplexión por división de frecuencia y después veremos cómo se puede aplicar FDM a la fibra óptica (multiplexión por división de longitud de onda). Después nos enfocaremos en TDM y terminaremos con un sistema TDM avanzado que se usa para fibra óptica (SONET).

Multiplexión por división de frecuencia

La figura 2-31 muestra cómo utilizar FDM para multiplexar tres canales telefónicos de calidad de voz. Los filtros limitan el ancho de banda utilizable a cerca de 3000 Hz por canal de calidad de voz. Cuando se multiplexan muchos canales juntos, se asignan 4000 Hz a cada canal para mantenerlos bien separados. Primero se eleva la frecuencia de los canales de voz, cada uno en una

cantidad diferente, después de lo cual se pueden combinar, porque en ese momento no hay dos canales que ocupen la misma porción del espectro. Observe que aunque existen separaciones entre los canales (bandas de protección), hay cierta superposición entre canales adyacentes porque los filtros no tienen bordes bien definidos. Esta superposición significa que un pico fuerte en el borde de un canal se detectará en el adyacente como ruido no térmico.

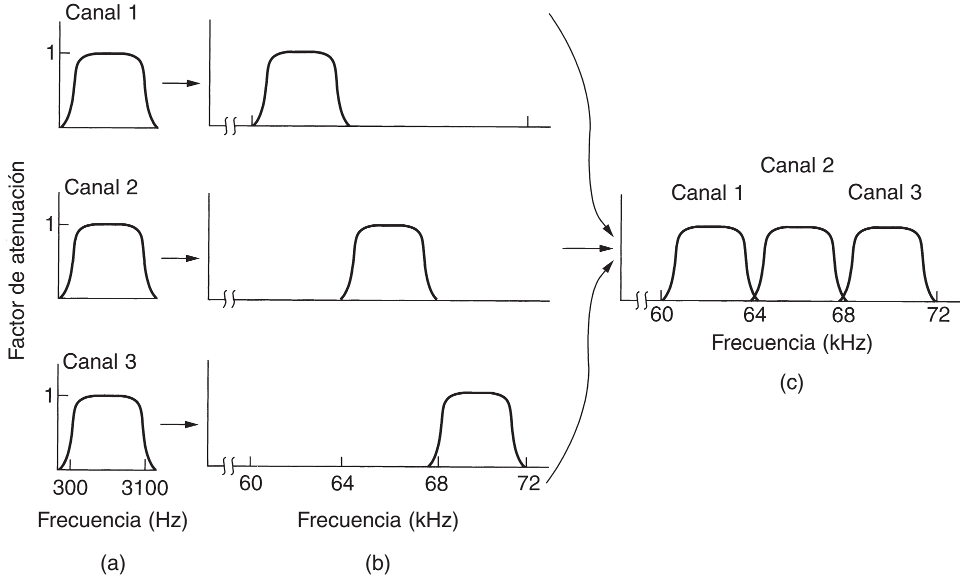


Figura 2-31. Multiplexión por división de frecuencia. (a) Los anchos de banda originales. (b) Incremento de frecuencia de los anchos de banda. (c) El canal multiplexado.

Los esquemas de FDM que se emplean en el mundo están normalizados hasta cierto punto. Un estándar muy difundido es el de 12 canales de voz a 4000 Hz multiplexados dentro de la banda de 60 a 108 kHz. Esta unidad se llama **grupo.** La banda de 12 a 60 kHz a veces se usa para otro grupo. Muchas empresas portadoras ofrecen un servicio de líneas alquiladas de 48 a 56 kbps que se basan en este grupo. Se pueden multiplexar cinco grupos (60 canales de voz) para formar un **supergrupo.** La siguiente unidad es el **grupo maestro,** que se compone de cinco supergrupos (en el estándar del CCITT) o de 10 supergrupos (en el sistema Bell). También existen otros estándares que llegan hasta 230,000 canales de voz.

Multiplexión por división de longitud de onda

Para los canales de fibra óptica se utiliza una variante de la multiplexión por división de frecuencia llamada **WDM** (**Multiplexión por División de Longitud de Onda).** En la figura 2-32 se muestran los principios básicos de la WDM en fibra. Aquí, cuatro fibras se juntan en un combinador óptico, cada una con su energía presente a diferentes longitudes de onda. Los cuatro haces se combinan en una sola fibra compartida para transmisión a un destino distante. En el extremo

distante, el haz se divide en tantas fibras como hayan entrado. Cada fibra saliente contiene un núcleo corto especialmente construido que filtra todas las longitudes de onda, excepto una. Las señales resultantes pueden enrutarse a su destino o recombinarse en diferentes formas para transporte adicional multiplexado.

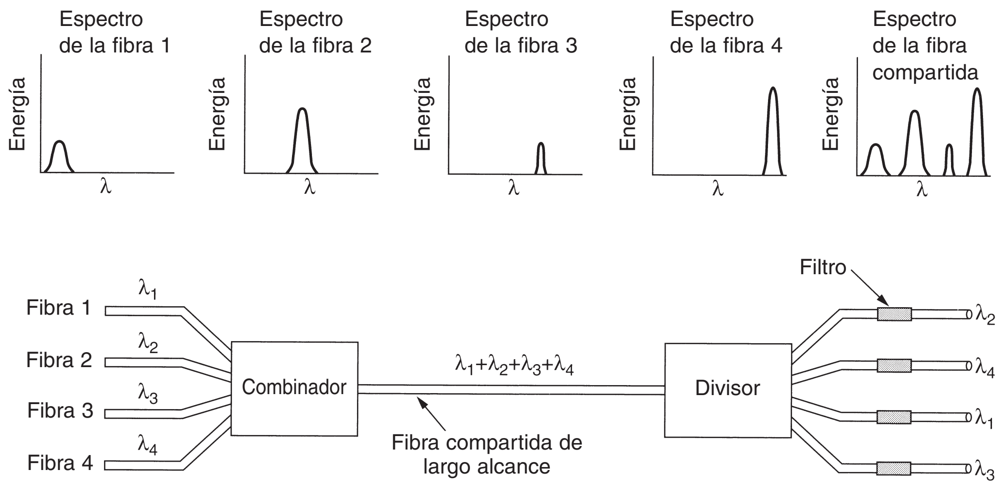


Figura 2-32. Multiplexión por división de longitud de onda.

En realidad, aquí nada es nuevo. Se trata simplemente de multiplexión por división de frecuencia a frecuencias muy altas. Siempre y cuando cada canal tenga su propio rango de frecuencia (es decir, longitud de onda), y todos los intervalos estén separados, se pueden multiplexar juntos en la fibra de largo alcance. La única diferencia con respecto a la FDM eléctrica es que un sistema óptico que usa una rejilla de difracción es totalmente pasivo y, por ello, muy confiable.

La tecnología WDM ha progresado de tal manera que ha dejado en vergüenza a la tecnología de computadoras. La WDM fue inventada en 1990. Los primeros sistemas comerciales tenían ocho canales, cada uno de los cuales era de 2.5 Gbps. En 1998, los sistemas con 40 canales de 2.5 Gbps ya estaban en el mercado. En 2001 había productos con 96 canales de 10 Gbps, con un total de 960 Gbps. Éste es suficiente ancho de banda como para transmitir 30 películas completas por segundo (en MPEG-2). Los sistemas con 200 canales ya están trabajando en el laboratorio. Cuando el número de canales es muy grande y las longitudes de onda están espaciadas entre sí de manera estrecha, por ejemplo a 0.1 nm, el sistema se conoce como **DWDM** (**WDM Densa**).

Cabe señalar que la razón por la que WDM es popular es que la energía de una sola fibra por lo general es de unos cuantos gigahertz debido a que en la actualidad es imposible convertir con mayor rapidez entre los medios óptico y eléctrico. Al ejecutar muchos canales en paralelo sobre diferentes longitudes de onda, el ancho de banda agregado se incrementa de manera lineal de acuerdo con el número de canales. Puesto que el ancho de banda de una sola banda de fibra es de alrededor de 25,000 GHz (vea la figura 2-6), teóricamente hay espacio para 2500 canales de 10 Gbps incluso a 1 bit/Hz (también son posibles tasas más altas).

Otro desarrollo novedoso es mediante amplificadores ópticos. Anteriormente, era necesario dividir todos los canales cada 100 km y convertir cada uno en una señal eléctrica para una amplificación por separado antes de volver a convertirlos a ópticos y combinarlos. En la actualidad todos los amplificadores pueden regenerar toda la señal una vez cada 1000 km sin necesidad de múltiples conversiones óptico-eléctricas.

En el ejemplo de la figura 2-32 tenemos un sistema de longitud de onda fija. Los bits de la fibra entrante 1 van a la fibra saliente 3, los de la fibra entrante 2 van a la fibra saliente 1, etcétera. Sin embargo, es posible construir sistemas WDM conmutados. En un dispositivo de ese tipo los filtros de salida se pueden ajustar mediante interferómetros de Fabry-Perot o de Mach-Zehnder. Para mayor información acerca de WDM y su aplicación en la conmutación de paquetes en Internet, vea (Elmirghani y Mouftah, 2000; Hunter y Andonovic, 2000, y Listani y cols., 2001).

Multiplexión por división de tiempo

La tecnología WDM es excelente, pero aún hay mucho cable de cobre en el sistema telefónico, por lo tanto, regresemos a ese tema por un momento. Aunque FDM aún se utiliza sobre cables de cobre o canales de microondas, requiere circuitos analógicos y no es fácil hacerla con una computadora. En contraste, TDM puede manejarse por completo mediante dispositivos digitales y a ello se debe su popularidad en los últimos años. Desgraciadamente, sólo se puede utilizar para datos digitales. Puesto que los circuitos locales producen señales analógicas, se necesita una conversión de analógico a digital en la oficina central, en donde todos los circuitos locales individuales se juntan para combinarse en troncales.

A continuación analizaremos la forma en que las múltiples señales analógicas de voz se digitalizan y combinan en una sola troncal digital saliente. Los datos de cómputo que se envían a través de un módem también son analógicos, por lo que la siguiente descripción también se aplica a ellos. Las señales analógicas se digitalizan en la oficina central con un dispositivo llamado **codec** (codificador-decodificador), con lo que se produce una serie de números de 8 bits. El codec toma 8000 muestras por segundo (125 μseg/muestra) porque el teorema de Nyquist dice que esto es suficiente para capturar toda la información del ancho de banda de 4 kHz del canal telefónico. A una velocidad de muestreo menor, la información se perdería; a una mayor, no se ganaría información extra. Esta técnica se llama **PCM** (**Modulación por Codificación de Impulsos**)**.** La PCM es el corazón del sistema telefónico moderno. En consecuencia, virtualmente todos los intervalos de tiempo dentro del sistema telefónico son múltiplos de 125 μseg.

Cuando la transmisión digital empezó a surgir como una tecnología factible, el CCITT era incapaz de lograr un acuerdo respecto al estándar internacional para la PCM. En consecuencia, ahora se usan diversos esquemas incompatibles en diferentes países alrededor del mundo.

Un método muy utilizado en Estados Unidos y Japón es el de la portadora **T1,** descrito en la figura 2-33. (Técnicamente hablando, el formato se llama DS1 y la portadora se llama T1, pero aquí no haremos esa sutil distinción.) La portadora T1 consiste en 24 canales de voz que se multiplexan juntos. Por lo común, las señales analógicas se muestrean por asignación cíclica (en *round robin*), alimentando el flujo analógico resultante al codec en lugar de tener 24 codecs y después mezclar la salida digital. Cada uno de los 24 canales inserta, a la vez, 8 bits en el flujo de salida.

Siete bits son de datos y uno es de control, con lo que se obtienen 7 × 8000 = 56,000 bps de datos, y 1 × 8000 = 8000 bps de información de señalización por canal.

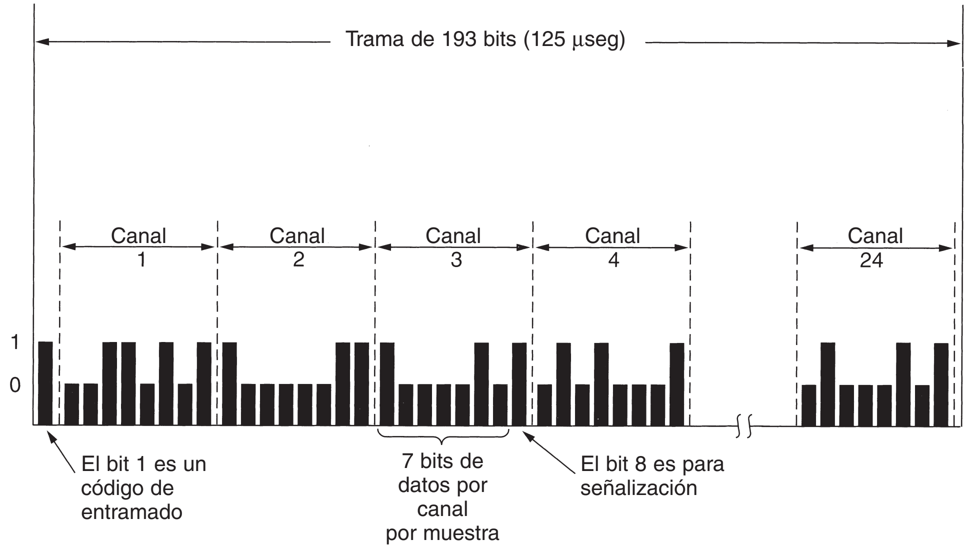


Figura 2-33. La portadora T1 (1.544 Mbps).

Una trama consiste en 24 × 8 = 192 bits más un bit extra para entramado, lo que da 193 bits cada 125 μseg. Esto produce una tasa de transmisión de datos bruta de 1.544 Mbps. El bit número 193 se usa para sincronización de la trama y sigue el patrón 0101010101... Por lo general, el receptor verifica de manera continua este bit para asegurarse de que no ha perdido la sincronización. Si llegara a perder sincronía, el receptor puede esperar hasta detectar otra vez el patrón y volverse a sincronizar. Los clientes analógicos no pueden generar el patrón de bits porque corresponde a una onda senoidal a 4000 Hz, que sería filtrada. Desde luego, los clientes digitales pueden generar este patrón, pero hay poca probabilidad de que esté presente cuando la trama pierda sincronía. Cuando se utiliza un sistema T1 exclusivamente para datos, sólo 23 de los canales llevan datos. El vigésimo cuarto lleva un patrón especial de sincronización que permite la recuperación rápida en caso de que la trama pierda sincronía.

Cuando el CCITT por fin llegó a un acuerdo, sintió que 8000 bps de información de señalización era demasiado, de modo que su estándar de 1.544 Mbps se basa en un elemento de datos de 8 bits en lugar de 7; es decir, la señal analógica se cuantiza en 256 niveles discretos en lugar de 128. Hay dos variantes (incompatibles). En la **señalización por canal común,** el bit extra (que se anexa al final y no al principio de la trama de 193 bits) adopta los valores 10101010... en las tramas nones y contiene información de señalización para todos los canales de las tramas pares.

En la otra variante, la **señalización por canal asociado,** cada canal tiene su propio subcanal privado de señalización. Se establece un subcanal privado asignando uno de los ocho bits de usuario

de cada sexta trama a funciones de señalización, así que cinco de cada seis muestras tienen 8 bits de ancho y la otra sólo tiene 7. El CCITT también recomendó una portadora PCM a 2.048 Mbps llamada **E1.** Esta portadora empaca 32 muestras de datos de 8 bits en la trama básica de 125 μseg. Treinta de los canales se usan para información y dos para señalización. Cada grupo de cuatro tramas proporciona 64 bits de señalización, la mitad de los cuales se usa para señalización por canal asociado y el resto se usa para sincronización de tramas o se reserva para que cada país los use como quiera. Fuera de Norteamérica y Japón, se utiliza la portadora E1 de 2.048 Mbps en lugar de la T1.

Una vez que la señal de voz se digitaliza, es tentador tratar de aplicar técnicas estadísticas para reducir la cantidad de bits necesarios por canal. Estas técnicas no sólo son apropiadas para codificar la voz, sino también para digitalizar cualquier señal analógica. Todos los métodos de compactación se basan en el principio de que la señal cambia con relativa lentitud en comparación con la frecuencia de muestreo, de modo que mucha de la información en el nivel digital de 7 u 8 bits es redundante.

Un método llamado **modulación diferencial por codificación de impulsos** consiste en transmitir no la amplitud digitalizada sino la diferencia entre su valor actual y el previo. Puesto que los saltos de ±16 en una escala de 128 no son probables, podrían bastar 5 bits en lugar de 7. Si la señal llegara a saltar de manera alocada en forma ocasional, la lógica de codificación podría requerir varios periodos de muestreo para "recuperarse". En el caso de la voz, se puede ignorar el error que se introduce.

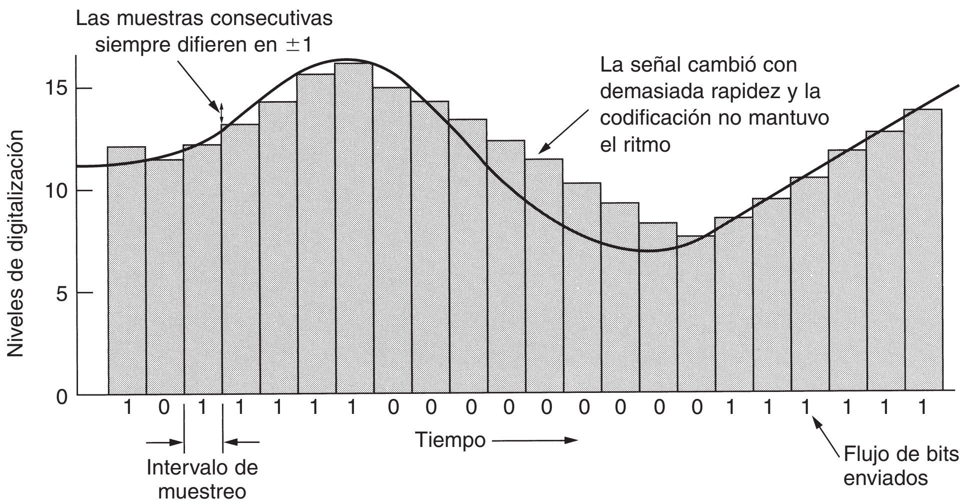


Figura 2-34. Modulación delta.

Una variante de este método de compactación requiere que cada valor muestreado difiera de su predecesor en +1 o −1. Bajo estas condiciones, se transmite un solo bit, que indica si la nueva

muestra está por arriba o por debajo de la anterior. En la figura 2-34 se ilustra esta técnica, llamada **modulación delta.** Al igual que todas las técnicas de compactación que suponen cambios pequeños de nivel entre muestras consecutivas, la codificación delta se puede meter en problemas si la señal cambia con demasiada rapidez, como se aprecia en la figura. Cuando esto sucede, se pierde información.

Una mejora a la PCM diferencial consiste en extrapolar algunos valores previos para predecir el siguiente valor y codificar a continuación la diferencia entre la señal real y la que se predice. Desde luego, el transmisor y el receptor deben utilizar el mismo algoritmo de predicción. A tales esquemas se les conoce como **codificación por predicción** y son útiles porque reducen el tamaño de los números que se codificarán y, por tanto, la cantidad de bits que se enviarán.

La multiplexión por división de tiempo permite que se multiplexen varias portadoras T1 en portadoras de orden más alto. La figura 2-35 muestra cómo se puede hacer esto. A la izquierda vemos que se multiplexan cuatro canales T1 en un canal T2. La multiplexión en T2 y superiores se hace bit por bit, en lugar de byte por byte, como en los 24 canales de voz que forman una trama T1. Cuatro flujos T1 a 1.544 Mbps deberían generar 6.176 Mbps, pero T2 en realidad es de 6.312 Mbps. Los bits adicionales sirven para entramar y para recuperar en caso de que la portadora pierda sincronía. T1 y T3 son utilizadas ampliamente por los clientes, mientras que T2 y T4 sólo se utilizan en el sistema telefónico mismo, por lo que no son muy conocidas.

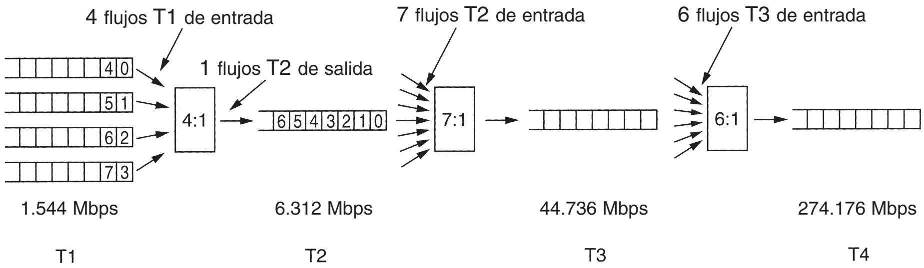


Figura 2-35. Multiplexión de flujos T1 en portadoras más altas.

En el siguiente nivel se combinan siete flujos T2 bit por bit que forman un flujo T3. A continuación se unen seis flujos T3 para formar un flujo T4. En cada paso se agrega una pequeña sobrecarga para entramado y recuperación en caso de que la sincronización entre el emisor y el receptor se pierda.

Así como existe un desacuerdo en lo tocante a la portadora básica entre Estados Unidos y el resto del mundo, también hay desacuerdo respecto a cómo se ha de multiplexar en portadoras de ancho de banda más alto. El esquema de Estados Unidos de dar pasos de 4, 6 y 7 no pareció lógico a todo el mundo, de modo que el estándar del CCITT prescribe la multiplexión de cuatro flujos en uno en cada nivel. Además, los datos de entramado y de recuperación son diferentes entre el estándar de Estados Unidos y el del CCITT. La jerarquía del CCITT para 32, 128, 512, 2048 y 8192 canales funciona a velocidades de 2.048, 8.848, 34.304, 139.264 y 565.148 Mbps.

SONET/SDH

En los primeros días de la fibra óptica, cada compañía telefónica tenía su propio sistema óptico TDM patentado. Después de que AT&T se dividió en 1984, las compañías telefónicas locales se tuvieron que conectar a múltiples empresas portadoras de larga distancia, todas con diferentes sistemas ópticos TDM, de modo que se hizo obvia la necesidad de estandarización. En 1985, Bellcore, la división de investigación de las RBOCs, empezó a trabajar en un estándar llamado **SONET** (**Red Óptica Síncrona**). Más tarde, el CCITT se unió al esfuerzo, lo que dio como resultado que en 1989 se produjera un estándar SONET y un conjunto de recomendaciones paralelas del CCITT (G.707, G.708 y G.709). A las recomendaciones del CCITT se les llama **SDH** (**Jerarquía Digital Síncrona**) pero difieren de SONET sólo en detalles menores. Virtualmente todo el tráfico telefónico de larga distancia en Estados Unidos y una buena parte del mismo en los demás países tiene ahora troncales que funcionan con SONET en la capa física. Si desea información adicional, vea (Bellamy, 2000; Goralski, 2000, y Shepard, 2001).

El diseño de SONET tuvo cuatro objetivos principales. Antes que nada, SONET tenía que hacer posible la interconexión de diferentes operadores telefónicos. El logro de este objetivo requirió que se definiera un estándar de señalización con respecto a la longitud de onda, la temporización, la estructura del entramado, etcétera.

Segundo, se necesitaron medidas para unificar los sistemas digitales estadounidense, europeo y japonés, todos los cuales se basaban en canales PCM de 64 kbps, pero combinados en formas diferentes (e incompatibles).

Tercero, SONET tenía que proporcionar un mecanismo para multiplexar varios canales digitales. En el momento en que se creó SONET, la portadora digital de mayor velocidad que se usaba ampliamente en Estados Unidos era la T3, a 44.736 Mbps. La T4 ya se había definido, pero no se utilizaba mucho, y todavía no se había definido nada por encima de la velocidad de T4. Parte de la misión de SONET era continuar la jerarquía a gigabits/seg y más allá. También se necesitaba una forma estándar de multiplexar canales más lentos en un solo canal SONET.

Cuarto, SONET tenía que proporcionar apoyo para las operaciones, la administración y el mantenimiento (OAM). Los sistemas anteriores no hacían esto muy bien.

Una decisión temprana fue convertir a SONET en un sistema TDM tradicional, con todo el ancho de banda de la fibra dedicado a un canal que contuviera ranuras de tiempo para los distintos subcanales. Como tal, SONET es un sistema síncrono, controlado por un reloj maestro con una precisión de alrededor de 1 parte en 10^9. En una línea SONET, los bits se envían a intervalos de suma precisión, controlados por el reloj maestro. Cuando posteriormente se propuso que la conmutación de celdas fuera la base de ATM, el hecho de que permitiera la llegada de celdas a intervalos irregulares le confirió la etiqueta de Modo de Transferencia *Asíncrona* para contrastarlo con el funcionamiento síncrono de SONET. Con este último, el emisor y el remitente están atados a un reloj común; con ATM no lo están.

La trama básica de SONET es un bloque de 810 bytes que se emite cada 125 μseg. Puesto que SONET es síncrona, las tramas se emiten haya o no datos útiles que enviar. La velocidad de 8000 tramas/seg coincide perfectamente con la tasa de muestreo de los canales PCM que se utilizan en todos los sistemas de telefonía digital.

Las tramas de 810 bytes de SONET se pueden describir mejor como un rectángulo de bytes de 90 columnas de ancho por nueve filas de alto. De este modo, $8 \times 810 = 6480$ bits se transmiten 8000 veces por segundo, lo que da una tasa de datos bruta de 51.84 Mbps. Éste es el canal básico de SONET y se llama **STS-1** (**Señal Síncrona de Transporte 1**). Todas las troncales de SONET son múltiplos de STS-1.

Las primeras tres columnas de cada trama se reservan para información de administración del sistema, como se ilustra en la figura 2-36. Las primeras tres filas contienen el encabezado de sección (*section overhead*); las siguientes seis contienen el encabezado de línea (*line overhead*). El encabezado de sección se genera y verifica al comienzo y al final de cada sección, mientras que el encabezado de línea se genera y verifica al comienzo y al final de cada línea.

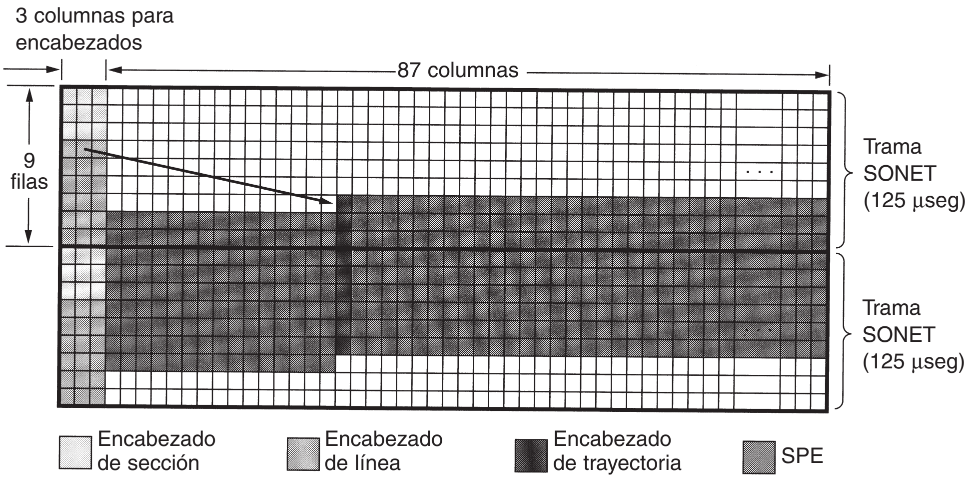


Figura 2-36. Dos tramas SONET consecutivas.

Un transmisor SONET envía tramas consecutivas de 810 bytes, sin huecos entre ellas, incluso cuando no hay datos (en cuyo caso envía datos ficticios). Todo lo que el receptor ve es un flujo continuo de bits, de modo que, ¿cómo sabe dónde comienza cada trama? La respuesta es que los dos primeros bytes de cada trama contienen un patrón fijo que el receptor busca. Si lo encuentra en el mismo lugar en una gran cantidad de tramas consecutivas, asume que está sincronizado con el emisor. En teoría, por lo general un usuario puede insertar este patrón en la carga útil, pero en la práctica no es posible hacer esto debido al multiplexado de múltiples usuarios que se realiza en la misma trama, entre otras razones.

Las 87 columnas restantes contienen $87 \times 9 \times 8 \times 8000 = 50.112$ Mbps de datos de usuario. Sin embargo, los datos de usuario, llamados **SPE** (**Contenedor o Sobre de Carga Útil Síncrona**), no siempre empiezan en la fila 1, columna 4. La SPE puede empezar en cualquier parte dentro de la trama. La primera fila del encabezado de línea contiene un apuntador al primer byte. La primera columna de la SPE es del encabezado de trayectoria (es decir, el encabezado para el protocolo de la subcapa de la trayectoria de extremo a extremo).

La facultad de que la SPE empiece en cualquier lugar dentro de la trama SONET o incluso abarque dos tramas, como se muestra en la figura 2-36, confiere una flexibilidad adicional al sistema. Por ejemplo, si una carga útil llega a la fuente mientras se está construyendo una trama SONET ficticia, se puede insertar en la trama actual, en lugar de retenerla hasta el inicio de la siguiente.

En la figura 2-37 se muestra la jerarquía de multiplexión de SONET. Se definieron tasas de STS-1 a STS-192. La portadora óptica que corresponde a cada STS-*n* se llama OC-*n*, pero es la misma bit por bit, excepto por un cierto reordenamiento de bits necesario para la sincronización. Los nombres de SDH son diferentes y empiezan en OC-3 porque los sistemas basados en el CCITT no tienen una tasa de transmisión cercana a los 51.84 Mbps. La portadora OC-9 está presente porque se aproxima mucho a la velocidad de una de las principales troncales de alta velocidad que se usan en Japón. OC-18 y OC-36 se utilizan en Japón. La tasa de datos bruta incluye todos los encabezados. La tasa de datos de SPE excluye los encabezados de línea y de sección. La tasa de datos de usuario excluye todos los encabezados y cuenta solamente las 86 columnas disponibles para la carga útil.

SONET		SDH	Tasa de datos (Mbps)		
Eléctrica	Óptica	Óptica	Bruta	SPE	De usuario
STS-1	OC-1		51.84	50.112	49.536
STS-3	OC-3	STM-1	155.52	150.336	148.608
STS-9	OC-9	STM-3	466.56	451.008	445.824
STS-12	OC-12	STM-4	622.08	601.344	594.432
STS-18	OC-18	STM-6	933.12	902.016	891.648
STS-24	OC-24	STM-8	1244.16	1202.688	1188.864
STS-36	OC-36	STM-12	1866.24	1804.032	1783.296
STS-48	OC-48	STM-16	2488.32	2405.376	2377.728
STS-192	OC-192	STM-64	9953.28	9621.504	9510.912

Figura 2-37. Tasas de multiplexión de SONET y SDH.

Por cierto, cuando una portadora, como la OC-3, no se multiplexa, sino que conduce datos de una fuente única, se agrega la letra *c* (de concatenado) a la designación, de modo que OC-3 indica una portadora de 155.52 Mbps consistente en tres portadoras OC-1 independientes, pero OC-3c indica un flujo de datos de una sola fuente a 155.52 Mbps. Los tres flujos OC-1 dentro de un flujo OC-3c se entrelazan por columnas, primero la columna 1 del flujo 1, a continuación la columna 1 del flujo 2, después la columna 1 del flujo 3 seguida de la columna 2 del flujo 1, y así sucesivamente, lo que produce una trama de 270 columnas de ancho y 9 filas de profundidad.

2.5.5 Conmutación

Desde el punto de vista de un ingeniero de telefonía ordinario, el sistema telefónico se divide en dos partes: planta externa (los circuitos locales y troncales, puesto que están fuera de las oficinas de conmutación) y planta interna (los conmutadores, que están dentro de las oficinas de

conmutación). Sólo hemos visto la planta externa. Llegó el momento de examinar la planta interna.

En la actualidad se utilizan dos técnicas de conmutación diferentes: conmutación de circuitos y conmutación de paquetes. A continuación presentaremos una breve introducción a cada una de ellas. Después veremos con detalle la conmutación de circuitos, porque así es como trabaja el sistema telefónico actual. Más adelante, en capítulos posteriores, examinaremos a fondo la conmutación de paquetes.

Conmutación de circuitos

Cuando usted o su computadora hacen una llamada telefónica, el equipo de conmutación del sistema telefónico busca una trayectoria física que vaya desde su teléfono al del receptor. Esta técnica se llama **conmutación de circuitos** y se muestra de manera esquemática en la figura 2-38(a). Cada uno de los seis rectángulos representa una oficina de conmutación de la empresa portadora (oficina central, oficina interurbana, etcétera). En este ejemplo, cada oficina tiene tres líneas de entrada y tres de salida. Cuando una llamada pasa por una oficina de conmutación, se establece una conexión física (en forma conceptual) entre la línea por la que llegó la llamada y una de las líneas de salida, lo que se representa mediante las líneas punteadas.

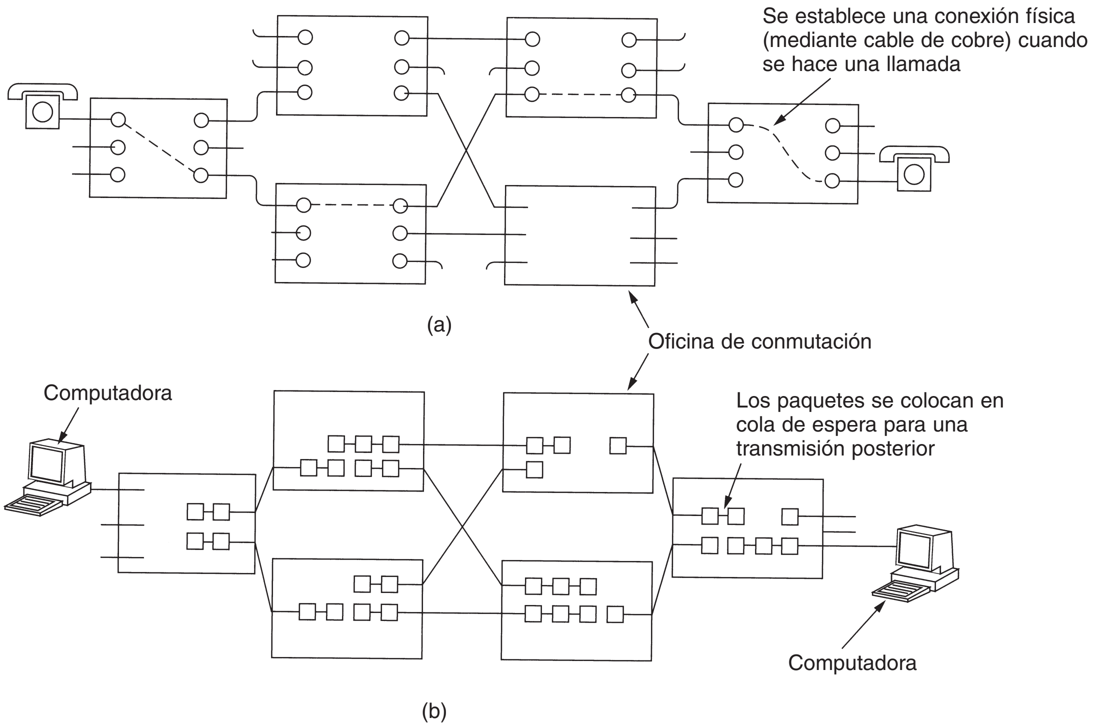


Figura 2-38. (a) Conmutación de circuitos. (b) Conmutación de paquetes.

En los primeros días del teléfono, se establecía la conexión cuando el operador conectaba un cable puenteador en los enchufes de entrada y de salida. Por cierto, existe una pequeña y sorprendente historia asociada a la invención del equipo de conmutación automática de circuitos: lo inventó el dueño de una funeraria del siglo XIX, un hombre llamado Almon B. Strowger. Poco después de que se inventara el teléfono, cuando alguien moría, alguno de los deudos llamaba a la operadora del pueblo y decía: “Por favor, comuníqueme con una funeraria”. Desgraciadamente para el señor Strowger, había dos funerarias en el pueblo, y la esposa del dueño de la otra era la operadora de teléfonos. Strowger pronto se dio cuenta de que si no inventaba el equipo de conmutación telefónica automática iba a quedar en bancarrota, así que eligió la primera opción. Durante casi 100 años, el equipo de conmutación de circuitos empleado en todo el mundo se conoció como el aparato de Strowger. (La historia no registra si la ahora desempleada operadora de conmutador obtuvo trabajo como operadora de información, contestando preguntas como: ¿Me da por favor el número de una funeraria?)

Desde luego, el modelo que se muestra en la figura 2-39(a) está altamente simplificado, porque partes de la trayectoria de “cobre” entre los dos teléfonos pueden ser, de hecho, enlaces de microondas en los cuales se multiplexan miles de llamadas. Sin embargo, la idea básica es válida: una vez que se ha establecido una llamada, existe una trayectoria dedicada entre ambos extremos y continuará existiendo hasta que termine la llamada.

La alternativa a la conmutación de circuitos es la conmutación de paquetes, que se muestra en la figura 2-38(b). Con esta tecnología, los paquetes individuales se envían conforme se necesite, y no se les asigna por adelantado ninguna trayectoria dedicada.

Una propiedad importante de la conmutación de circuitos es la necesidad de establecer una trayectoria de un extremo a otro *antes* de que se pueda enviar cualquier dato. El tiempo que transcurre entre que se termina de marcar y que el timbre comienza a sonar puede ser fácilmente de 10 seg, y más en las llamadas de larga distancia o internacionales. Durante este intervalo de tiempo, el sistema telefónico busca una trayectoria de cobre, como se muestra en la figura 2-39(a). Observe que antes de que pueda comenzar la transmisión de datos, la señal de petición de llamada se debe propagar hasta el destino y se debe confirmar su recepción. En muchas aplicaciones de computadora (por ejemplo, la verificación de crédito en un punto de venta), los tiempos de establecimiento largos son indeseables.

Al existir una trayectoria de cobre entre las partes en comunicación, una vez que se termina de establecer, el único retardo de los datos es el tiempo de propagación de la señal electromagnética, alrededor de 5 mseg por cada 1000 km. Otra ventaja de la trayectoria establecida es que no hay peligro de congestión; es decir, una vez que la llamada entra, no hay posibilidad de obtener una señal de ocupado, aunque podría obtener una antes de establecer la conexión debido a la falta de capacidad de conmutación o de troncal.

Conmutación de mensajes

Una estrategia de conmutación alterna es la **conmutación de mensajes** que se muestra en la figura 2-39(b). Cuando se usa esta forma de conmutación, no se establece por adelantado una trayectoria de cobre física entre el emisor y el receptor. En cambio, cuando el emisor tiene un blo-

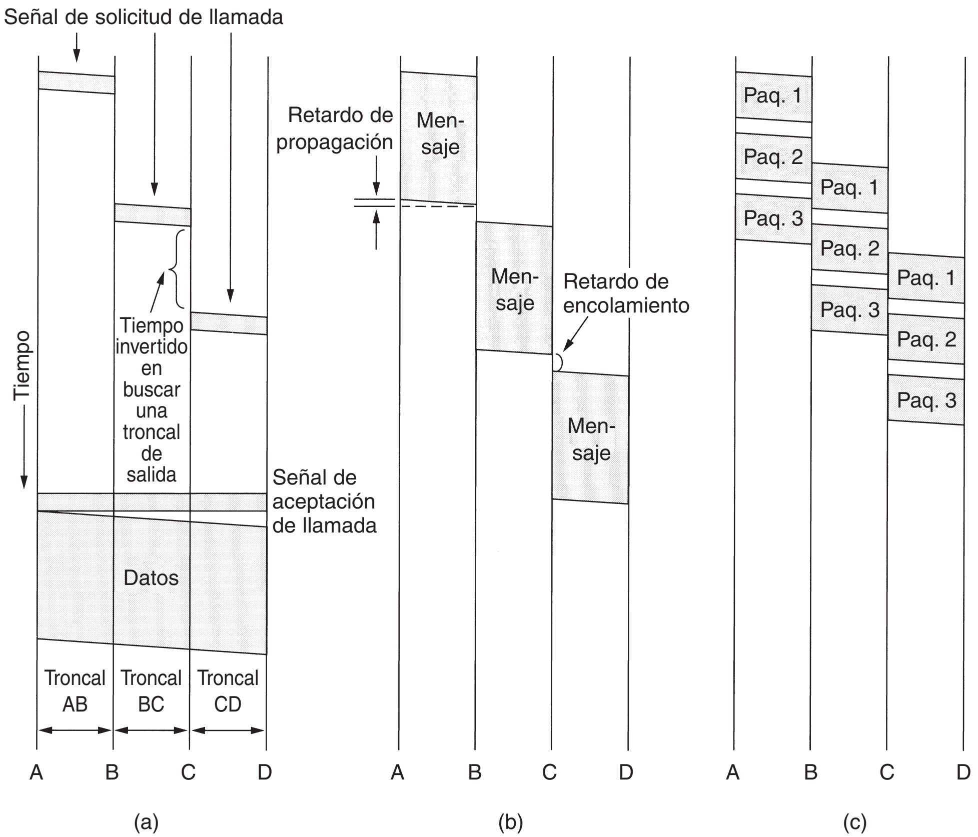


Figura 2-39. Tiempos de los eventos en (a) conmutación de circuitos, (b) conmutación de mensajes, (c) conmutación de paquetes.

que de datos para enviar, éste se almacena en la primera oficina de conmutación (es decir, enrutador) y después se reenvía, un salto a la vez. Cada bloque se recibe en su totalidad, se inspecciona en busca de errores y, después, se retransmite. Una red que utiliza esta técnica se conoce como red de **almacenamiento y reenvío** (*store and forward*), como se mencionó en el capítulo 1.

Los primeros sistemas de telecomunicación electromecánicos usaban conmutación de mensajes para enviar telegramas. El mensaje se perforaba en cinta de papel (fuera de línea) en la oficina emisora y después se leía y transmitía por una línea de comunicación a la siguiente oficina a lo largo del recorrido, donde se perforaba en cinta de papel. Allí, una operadora arrancaba la cinta de papel y la insertaba en una de las muchas lectoras de cinta, una por cada troncal de salida. Tal oficina de conmutación se llamaba **oficina de arrancado de cinta de papel.** La cinta de papel desapareció hace mucho tiempo y la conmutación de mensajes ya no se utiliza, por lo que ya no la analizaremos en este libro.

Conmutación de paquetes

Con la conmutación de mensajes, no hay límite para el tamaño de los bloques, lo que significa que los enrutadores (en un sistema moderno) deben tener discos para almacenar en forma temporal los bloques grandes. También significa que un solo bloque puede acaparar una línea de enrutador a enrutador durante minutos, lo que hace inútil la conmutación de mensajes para el tráfico interactivo. Con el fin de resolver estos problemas se inventó la **conmutación de paquetes,** como se describió en el capítulo 1. Las redes de conmutación de paquetes establecen un límite superior al tamaño del bloque, lo que permite almacenar los paquetes en la memoria principal del enrutador y no en el disco. Al asegurarse de que ningún usuario pueda monopolizar una línea de transmisión durante mucho tiempo (milisegundos), las redes de conmutación de paquetes pueden manejar tráfico interactivo. En la figura 2-39 (b) y (c) se muestra una ventaja adicional de la conmutación de paquetes sobre la conmutación de mensajes: el primer paquete de un mensaje de varios paquetes se puede reenviar antes de que el segundo haya llegado por completo, lo que reduce el retardo y mejora el rendimiento. Por estas razones, las redes de computadoras por lo general son de conmutación de paquetes, ocasionalmente de conmutación de circuitos y nunca de conmutación de mensajes.

La conmutación de circuitos y la de paquetes difieren en muchos aspectos. Para empezar, la conmutación de circuitos requiere que un circuito se establezca de extremo a extremo antes de que comience la comunicación. La conmutación de paquetes no requiere un establecimiento previo. El primer paquete puede simplemente enviarse tan pronto como esté disponible.

El resultado del establecimiento de conexión mediante la conmutación de circuito es la reserva de ancho de banda que se realiza desde el emisor hasta el receptor. Todos los paquetes siguen esta trayectoria. Entre otras propiedades, el hecho de que todos los paquetes sigan la misma trayectoria significa que no llegarán en desorden a su destino. Con la conmutación de paquetes no hay trayectoria, por lo que diferentes paquetes pueden seguir trayectorias distintas, dependiendo de las condiciones de la red en el momento en el que se enviaron. Pueden llegar en desorden.

La conmutación de paquetes es más tolerante a las fallas que la conmutación de circuitos. De hecho, ésa es la razón por la cual se inventó. Si falla la conmutación, todos los circuitos que la están utilizando se cancelan y no se puede enviar nada más a través de ellos. Con la conmutación de paquetes, los paquetes pueden enrutarse evitando a los conmutadores averiados.

Establecer con antelación una trayectoria también abre la posibilidad de reservar ancho de banda con antelación. Si se reserva ese ancho de banda, cuando un paquete llega, puede enviarse de manera inmediata a través de él. Con la conmutación de paquetes no se reserva ningún ancho de banda, por lo que los paquetes podrían tener que esperar su turno para ser reenviados.

Reservar ancho de banda con antelación significa que cuando llegue un paquete no habrá congestión (a menos de que lleguen más paquetes que los esperados). Por otra parte, cuando se intenta establecer un circuito, el intento puede fallar debido a la congestión. Por lo tanto, la congestión puede ocurrir en diversas ocasiones con la conmutación de circuitos (al momento del establecimiento) y con la de paquetes (cuando el paquete se envía).

Si un circuito se ha reservado para un usuario en particular y no hay tráfico que enviar, el ancho de banda de ese circuito se desperdicia. No se puede utilizar para otro tráfico. La conmutación de paquetes no desperdicia ancho de banda y, por lo tanto, es más eficiente desde el punto de vista del sistema. Entender esta compensación es crucial para entender la diferencia entre la con-

mutación de circuitos y la de paquetes. La compensación está entre un servicio garantizado con desperdicio de recursos contra un servicio no garantizado pero sin desperdicio de recursos.

La conmutación de paquetes utiliza transmisión de almacenamiento y reenvío. Un paquete se almacena en la memoria del enrutador y luego se reenvía al siguiente enrutador. Con la conmutación de paquetes los bits simplemente fluyen de manera continua a través del cable. La técnica de almacenamiento y reenvío agrega retardo.

Otra diferencia es que la conmutación de circuitos es totalmente transparente. El emisor y el receptor pueden usar cualquier tasa de transmisión, formato o método de entramado de bits que quieran. La empresa portadora no lo sabe ni le interesa. Con la conmutación de paquetes la empresa portadora determina los parámetros básicos. Una analogía burda sería comparar un camino con una vía de tren. En el primero, el usuario determina el tamaño, la velocidad y la naturaleza del vehículo; en la vía del tren esto lo hace el prestador de servicios. Esta transparencia es la que hace posible que coexistan voz, datos y fax dentro del sistema telefónico.

Una diferencia final entre la conmutación de circuitos y la de paquetes es el algoritmo de cobro. En la conmutación de circuitos, el cobro se ha basado históricamente en la distancia y el tiempo. En el caso de los teléfonos móviles, la distancia, por lo general, no es importante, excepto cuando se trata de llamadas internacionales, y el tiempo tiene poca importancia (por ejemplo, un plan de llamadas con 2000 minutos libres cuesta más que uno con 1000 minutos libres y algunas veces las llamadas de noche o de fin de semana son más baratas de lo normal). En el caso de la conmutación de paquetes, el tiempo de conexión no es un problema, pero con frecuencia el volumen del tráfico sí lo es. Por lo general, los ISPs (proveedores de servicios de Internet) cargan a los usuarios domésticos una tarifa mensual porque es más sencillo y sus clientes pueden entender este modelo con mayor facilidad, pero las empresas portadoras de red dorsal realizan cargos a las redes regionales con base en el volumen de su tráfico. Las diferencias se resumen en la figura 2-40.

Elemento	Conmutación de circuitos	Conmutación de paquetes
Establecimiento de llamada	Requerido	No es necesario
Trayectoria física detallada	Sí	No
Cada paquete puede seguir la misma trayectoria	Sí	No
Los paquetes llegan en orden	Sí	No
Es una falla de conmutación fatal	Sí	No
Ancho de banda disponible	Fijo	Dinámico
Cuándo puede haber congestión	Durante el establecimiento	En cada paquete
Ancho de banda potencialmente desperdiciado	Sí	No
Transmisión de almacenamiento y reenvío	No	Sí
Transparencia	Sí	No
Cargos	Por minuto	Por paquete

Figura 2-40. Comparación de redes de conmutación de circuitos y conmutación de paquetes.

Tanto la conmutación de circuitos como la de paquetes son tan importantes que regresaremos a ellas dentro de poco y describiremos en detalle las diversas tecnologías que se usan.

2.6 EL SISTEMA TELEFÓNICO MÓVIL

El sistema telefónico tradicional (incluso aunque algún día llegará a utilizar la fibra de extremo a extremo de múltiples gigabits) no podrá satisfacer un grupo creciente de usuarios: personas en movimiento. Los usuarios ahora esperan realizar llamadas telefónicas desde aviones, automóviles, albercas y mientras corren en el parque. Dentro de algunos años también esperarán poder enviar correo electrónico y navegar por Web desde cualquiera de las ubicaciones antes mencionadas, entre muchas otras cosas. En consecuencia, hay demasiado interés en la telefonía inalámbrica. En las siguientes secciones estudiaremos este tema con mayor detalle.

Los teléfonos inalámbricos se dividen en dos categorías básicas: **teléfonos inalámbricos** y teléfonos móviles (algunas veces llamados **teléfonos celulares).** Los primeros son dispositivos que consisten en una estación base y un teléfono que se venden en conjunto para utilizarse dentro de una casa. Éstos nunca se utilizan para conectividad de redes, por lo que no los trataremos más. En su lugar nos concentraremos en el sistema móvil, que se utiliza para la comunicación de datos y voz de área amplia.

Los **teléfonos móviles** han pasado por tres generaciones distintas, con tecnologías diferentes:

1. Voz analógica.
2. Voz digital.
3. Voz y datos digitales (Internet, correo electrónico, etcétera).

Aunque la mayor parte de nuestro análisis se concentra en la tecnología de estos sistemas, es interesante mencionar cómo es que las decisiones políticas y de publicidad pueden tener un gran impacto. El primer sistema móvil fue diseñado en Estados Unidos por AT&T y regulado por la FCC. Como resultado, Estados Unidos tenía un solo sistema (analógico), y un teléfono celular comprado en California también funcionaba en Nueva York. En contraste, cuando el sistema móvil apareció en Europa, cada país diseñó su propio sistema, lo cual fue un fracaso.

Europa aprendió de su error y cuando aparecieron los sistemas digitales, los PTTs a cargo del gobierno se unieron y estandarizaron un solo sistema (GSM), por lo que cualquier teléfono móvil europeo funcionaría en cualquier lugar de Europa. En ese entonces, Estados Unidos decidió que el gobierno no debería estar en el negocio de la estandarización, por lo que dejó la cuestión de los sistemas digitales al mercado. Esta decisión resultó en diferentes fabricantes que producían diferentes tipos de teléfonos móviles. En consecuencia, Estados Unidos ahora tiene funcionando dos principales sistemas telefónicos móviles digitales incompatibles (además de otro menor).

A pesar de la ventaja inicial que tenía Estados Unidos, la posesión y el uso de teléfonos móviles en Europa ahora es mayor que en Estados Unidos. El hecho de tener un solo sistema para toda Europa es una de las razones, pero hay más. Una segunda parte en la que Europa y Estados Unidos difieren es en la cuestión de los números telefónicos. En Estados Unidos los teléfonos móviles están mezclados con los teléfonos (fijos) normales. Por lo tanto, no hay forma de que la persona que llama vea si, digamos, (212)234-5678 es el número de un teléfono fijo (barato o de

llamada gratis) o uno de teléfono móvil (llamada costosa). Para que la gente no se asustara de utilizar los teléfonos móviles, las compañías telefónicas decidieron que el dueño de un teléfono móvil pague por las llamadas entrantes. En consecuencia, muchas personas dudaron en comprar un teléfono móvil por miedo a terminar con una gran cuenta por pagar sólo por recibir llamadas. En Europa, los números de los teléfonos móviles tienen un código de área especial (parecido a los números 800 y 900) por lo que se pueden reconocer al instante. Como resultado la regla común de "el que llama paga" también se aplica a los teléfonos en Europa (excepto en las llamadas internacionales en las que el costo se divide).

Un tercer aspecto que ha tenido un gran impacto en la adopción es el amplio uso de los teléfonos móviles prepagados en Europa (hasta de 75% en algunas áreas). Pueden comprarse en muchas tiendas de la misma manera que un radio; simplemente se pagan. Se precargan con, digamos, 20 o 50 euros y pueden recargarse (utilizando un código de PIN secreto) cuando el saldo se acaba. En consecuencia, prácticamente todos los adolescentes y muchos niños de Europa tienen teléfonos móviles (por lo general, prepagados), y de esta manera sus padres pueden localizarlos sin el peligro de que el niño termine con una cuenta enorme. Si el teléfono móvil sólo se utiliza de vez en cuando, su uso es esencialmente libre debido a que no hay un cargo mensual ni uno por llamadas entrantes.

2.6.1 Teléfonos móviles de primera generación: voz analógica

Ya es suficiente sobre los aspectos políticos y de marketing de los teléfonos celulares. Ahora examinemos a la tecnología, comenzando con el sistema más antiguo. Los radioteléfonos móviles se utilizaban de forma esporádica para comunicación marítima y militar durante las primeras décadas del siglo XX. En 1946, el primer sistema de teléfonos instalado en autos se construyó en St. Louis. Este sistema utilizaba un solo transmisor grande colocado en la parte superior de un edificio y tenía un solo canal que servía para enviar y recibir. Para hablar, el usuario tenía que oprimir un botón que habilitaba el transmisor e inhabilitaba el receptor. Tales sistemas, conocidos como **sistemas de oprimir para hablar,** se instalaron en algunas ciudades desde finales de la década de 1950. El radio de banda civil (CB), los taxis y las patrullas policiacas en programas de televisión a veces usan esta tecnología.

En la década de 1960 se instaló el **IMTS** (**Sistema Mejorado de Telefonía Móvil**)**.** También utilizaba un transmisor de alta potencia (200 watts), en la cima de una colina, pero tenía dos frecuencias, una para enviar y otra para recibir, por lo que el botón de oprimir para hablar ya no era necesario. Puesto que toda la comunicación desde los teléfonos móviles entraba por un canal diferente del que recibían los teléfonos emisores, los usuarios móviles no podían escucharse unos a otros (a diferencia del sistema de oprimir para hablar empleado en los taxis).

IMTs manejaba 23 canales dispersos desde 150 hasta 450 MHz. Debido al número tan pequeño de canales, los usuarios a veces tenían que esperar bastante tiempo antes de obtener el tono de marcar. Además, debido a la gran potencia del transmisor en la cima de la colina, los sistemas adyacentes tenían que estar alejados varios cientos de kilómetros para evitar la interferencia. Considerando todo, el sistema no era práctico debido a su capacidad limitada.

Sistema Avanzado de Telefonía Móvil

Todo cambió con **AMPS (Sistema Avanzado de Telefonía Móvil),** inventado por los Laboratorios Bell e instalado por primera vez en Estados Unidos en 1982. Este sistema también se utilizó en Inglaterra, donde se llamó TACS, y en Japón, donde se llamó MCS-L1. Aunque no es lo último en tecnología, lo analizaremos, pues muchas de sus propiedades fundamentales han sido heredadas por su sucesor digital, D-AMPS, con el fin de tener compatibilidad hacia atrás.

En todos los sistemas de telefonía móvil, una región geográfica se divide en **celdas,** razón por la cual los dispositivos se conocen como teléfonos celulares. En AMPS, las celdas normalmente tienen de 10 a 20 km de diámetro; en los sistemas digitales, las celdas son más pequeñas. Cada celda utiliza un conjunto de frecuencias que no es utilizada por ninguno de sus vecinos. La idea clave que confiere a los sistemas celulares más capacidad que todos los sistemas anteriores es el uso de celdas relativamente pequeñas y la reutilización de las frecuencias de transmisión en celdas cercanas (pero no adyacentes). Mientras que un sistema IMTS de 100 km de alcance puede tener una llamada en cada frecuencia, un sistema AMPS podría tener 100 celdas de 10 km en la misma área con 5 a 10 llamadas en cada frecuencia en celdas muy separadas. Por lo tanto, el diseño celular incrementa la capacidad del sistema en un orden de magnitud conforme las celdas se hacen más pequeñas en su área de cobertura. Además, al ser las celdas más pequeñas se necesita menor potencia, lo cual conduce a dispositivos más pequeños y económicos. Los teléfonos de bolsillo tienen una salida de 0.6 watts; los transmisores en los automóviles normalmente son de 3 watts, el máximo permitido por la FCC.

La idea de reutilizar frecuencias se ilustra en la figura 2-41(a). Por lo general, las celdas son casi circulares, pero es más fácil modelarlas como hexágonos. En la figura 2-41(a), las celdas son del mismo tamaño y están agrupadas en unidades de siete celdas. Cada letra indica un grupo de frecuencias. Observe que para cada conjunto de frecuencias hay un espacio de alrededor de dos celdas de ancho en el que esa frecuencia no se reutiliza, proporcionando buena separación y baja interferencia.

Encontrar localidades elevadas para colocar antenas de estación base es un problema importante. Este problema ha llevado a algunas empresas portadoras de telecomunicaciones a forjar alianzas con la Iglesia Católica Romana, puesto que ésta posee un número sustancial de sitios potenciales para antenas en todo el mundo, todos convenientemente bajo una administración única.

En un área en la que la cantidad de usuarios ha crecido tanto que el sistema está sobrecargado, la potencia se reduce y las celdas sobrecargadas se dividen en **microceldas** para permitir una mayor reutilización de las frecuencias, como se muestra en la figura 2-41 (a). Las compañías telefónicas algunas veces crean microceldas temporales, utilizando torres portables con enlaces de satélite, en eventos deportivos, conciertos de rock y otros lugares donde un gran número de usuarios móviles se congrega por algunas horas. Qué tan grandes deben ser las celdas es un tema complejo, y se trata en (Hac, 1995).

En el centro de cada celda está la estación base a la cual transmiten todos los teléfonos de la celda. La estación base consiste en una computadora y un transmisor/receptor conectado a una antena. En un sistema pequeño, todas las estaciones base se conectan a un mismo dispositivo llamado **MTSO (Oficina de Conmutación de Telefonía Móvil)** o **MSC (Centro de Conmutación Móvil).**

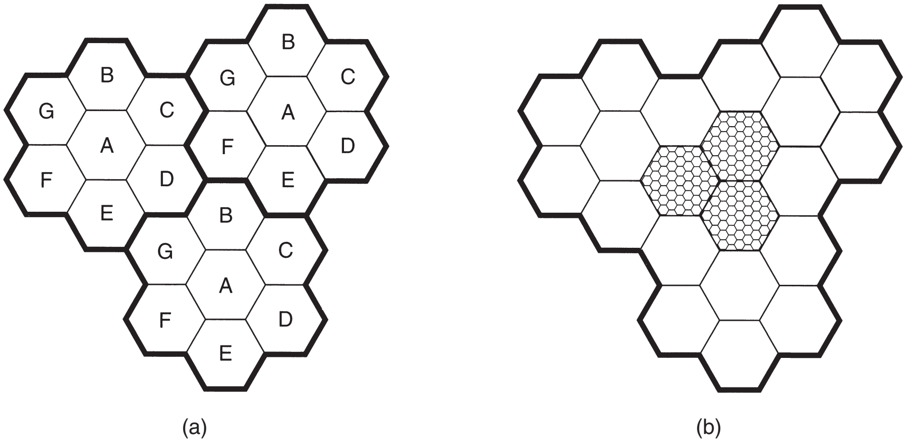


Figura 2-41. (a) Las frecuencias no se reutilizan en celdas adyacentes. (b) Para añadir más usuarios se pueden usar celdas más pequeñas.

En un sistema grande pueden ser necesarias varias MTSOs, las cuales se conectan a una MTSO de segundo nivel, y así sucesivamente. Las MTSOs son esencialmente oficinas centrales como en un sistema telefónico y, de hecho, están conectadas a por lo menos una oficina central del sistema telefónico. Las MTSOs se comunican con las estaciones base, con otras MTSOs y con la PSTN mediante una red de conmutación de paquetes.

En cualquier instante cada teléfono móvil está en una celda específica y bajo el control de la estación base de esa celda. Cuando un teléfono móvil sale de una celda, su estación base nota que la señal telefónica se desvanece o pregunta a todas las estaciones base circundantes cuánta potencia están recibiendo de ella. A continuación, la estación base transfiere la posesión a la celda que está recibiendo la señal más fuerte, esto es, la celda donde se localiza ahora el teléfono. Se informa entonces al teléfono cuál es su nueva base y, si está efectuando una llamada, se le pide que cambie a un nuevo canal (porque el antiguo no se reutiliza en ninguna celda adyacente). Este proceso, llamado **transferencia de celda** (*cell handoff*), tarda cerca de 300 mseg. La asignación de canal es realizada por la MTSO, que es el centro neurálgico del sistema. Las estaciones base sólo son retransmisoras de radio.

Las transferencias de celda pueden realizarse de dos maneras. En la **transferencia suave de celda** (*soft handoff*), el teléfono es adquirido mediante la nueva estación base antes de que las primeras terminen. De esta manera, no hay pérdida de continuidad. La desventaja es que el teléfono necesita estar disponible para sintonizar dos frecuencias al mismo tiempo (la anterior y la nueva). Ni los dispositivos de primera generación ni los de segunda pueden hacer esto.

En la **transferencia dura de celda** (*hard handoff*) la antigua estación base deja el teléfono antes de que la nueva lo adquiera. Si la nueva no puede adquirirlo (por ejemplo, debido a que no hay frecuencia disponible), la llamada se termina de manera abrupta. Los usuarios suelen notar esto, pero con el diseño actual es inevitable que suceda en ocasiones.

Canales

El sistema AMPS emplea 832 canales dúplex, cada uno compuesto por un par de canales símplex. Hay 832 canales de transmisión símplex desde 824 hasta 849 MHz, y 832 canales de recepción símplex desde 869 hasta 894 MHz. Cada uno de estos canales símplex es de 30 kHz de ancho; por lo tanto, AMPS usa FDM para separar los canales.

En la banda de 800 MHz, las ondas de radio son de cerca de 40 cm de largo y viajan en línea recta; son absorbidas por árboles y plantas y rebotan en el suelo y los edificios. Es posible que una señal enviada por un teléfono móvil llegue a la estación base por una trayectoria directa, pero también con un pequeño retardo después de rebotar en el suelo o en un edificio. Esto puede producir un efecto de eco o de distorsión de la señal (desvanecimiento de múltiples trayectorias). A veces es posible incluso oír una conversación distante que ha rebotado varias veces.

Los 832 canales se dividen en cuatro categorías:

1. Control (base a móvil) para administrar el sistema.
2. Localización (base a móvil) para avisar a usuarios móviles que tienen llamadas.
3. Acceso (bidireccional) para establecimiento de llamadas y asignación de canales.
4. Datos (bidireccional) para voz, fax o datos.

Veintiún canales se reservan para control, y están fijos dentro de un PROM en cada teléfono. Puesto que las mismas frecuencias no pueden reutilizarse en celdas cercanas, la cantidad real de canales de voz disponibles por célula es mucho menor que 832, normalmente cerca de 45.

Administración de llamadas

Cada teléfono móvil en AMPS tiene un número de serie de 32 bits y un número telefónico de 10 dígitos en su PROM. El número de teléfono se representa como un código de área de 3 dígitos, en 10 bits, y un número de suscriptor de 7 dígitos, en 24 bits. Cuando un teléfono se enciende, examina una lista preprogramada de 21 canales de control para encontrar la señal más potente.

A continuación, el teléfono difunde su número de serie de 32 bits y su número de teléfono de 34 bits. Al igual que toda la información de control de AMPS, este paquete se envía en forma digital varias veces y con código de corrección de errores, aunque los canales de voz mismos son analógicos.

Cuando la estación base oye el anuncio, avisa a la MTSO, la cual registra la existencia de su nuevo cliente y también informa a la MTSO local del cliente de su ubicación actual. Durante el funcionamiento normal, el teléfono móvil se vuelve a registrar cada 15 minutos aproximadamente.

Para hacer una llamada, un usuario móvil enciende el teléfono, teclea el número al que desea llamar y oprime el botón de Enviar. El teléfono envía entonces el número al que se va a llamar y su propia identidad por el canal de acceso. Si ocurre una colisión, lo intenta nuevamente más tarde.

Cuando la estación base recibe la petición, informa a la MTSO. Si el que llama es un cliente de la compañía de la MTSO (o uno de sus socios), la MTSO busca un canal desocupado para la llamada; si encuentra uno, el número de canal se envía de regreso por el canal de control. A continuación, el teléfono móvil se conecta en forma automática con el canal de voz seleccionado y espera hasta que la persona llamada levante el teléfono.

Las llamadas entrantes funcionan de forma diferente. Para empezar, todos los teléfonos desocupados escuchan continuamente el canal de aviso para detectar mensajes dirigidos a ellos. Cuando se hace una llamada a un teléfono móvil (ya sea desde un teléfono fijo o algún otro teléfono móvil), se envía un paquete a la MTSO local del destinatario de la llamada para averiguar dónde se encuentra. Luego se envía un paquete a la estación base de su celda actual, la cual realiza una difusión por el canal de aviso de la forma: “unidad 14, ¿está ahí?” A continuación el teléfono llamado responde con “Sí” por el canal de control. Enseguida, la base dice algo como: “unidad 14, tiene llamada por el canal 3”. En este punto, el teléfono llamado conmuta al canal 3 y empieza a timbrar.

2.6.2 Teléfonos móviles de segunda generación: voz digital

La primera generación de teléfonos móviles fue analógica; la segunda fue digital. Al igual que no hubo estandarización mundial en la primera generación, tampoco la hubo en la segunda. En la actualidad hay cuatro sistemas en uso: D-AMPS, GSM, CDMA y PDC. A continuación analizaremos las primeras tres. PDC sólo se utiliza en Japón y básicamente es un D-AMPS modificado para compatibilidad hacia atrás con el sistema analógico japonés de primera generación. A veces se utiliza el nombre **PCS** (**Servicios de Comunicaciones Personales**) para indicar el sistema de segunda generación (es decir, el digital). Originalmente denotaba un teléfono móvil que utilizaba la banda de 1900 MHz, pero en la actualidad esa distinción se emplea raramente.

D-AMPS—El Sistema Avanzado de Telefonía Móvil Digital

La segunda generación de los sistemas AMPS es **D-AMPS** y es completamente digital. Se describe en el estándar internacional IS-54 y en su sucesor IS-136. D-AMPS se diseñó con mucho cuidado para que pudiera coexistir con AMPS, a fin de que tanto los teléfonos móviles de primera generación como los de segunda pudieran funcionar de manera simultánea en la misma celda. En particular, D-AMPS utiliza los mismos canales a 30 kHz que AMPS y a las mismas frecuencias a fin de que un canal pueda ser analógico y los adyacentes, digitales. Dependiendo de la mezcla de teléfonos en las celdas, la MTSO de la celda determina cuáles canales son analógicos y cuáles digitales, y puede cambiar tipos de canales de manera dinámica conforme cambie la mezcla de canales en una celda.

Cuando D-AMPS fue introducido como un servicio, se puso disponible una nueva banda de frecuencia para manejar la carga esperada creciente. Los canales ascendentes estaban en el rango de 1850-1910 MHz, y los canales descendentes correspondientes estaban en el rango de 1930-1990 MHz, nuevamente en pares, como en AMPS. En esta banda, las ondas son de 16 cm de

longitud, por lo que una antena de onda estándar de ¼ es de sólo 4 cm de longitud, lo que da teléfonos más pequeños. Sin embargo, muchos teléfonos D-AMPS pueden utilizar tanto las bandas de 850-MHz como las de 1900-MHz para obtener una gama más amplia de canales disponibles.

En un teléfono móvil D-AMPS, la señal de voz capturada por el micrófono se digitaliza y comprime utilizando un modelo más refinado que los esquemas de modulación delta y de codificación de predicción que analizamos anteriormente. La compresión toma en cuenta propiedades del sistema de voz humano para obtener el ancho de banda de la codificación PCM estándar de 56 a 8 kbps o menos. La compresión se crea mediante un circuito llamado **vocoder** (Bellamy, 2000). La compresión se realiza en el teléfono, en lugar de en la estación base o en la oficina central, para reducir el número de bits que se envían a través del enlace de aire. Con la telefonía fija, no hay beneficio de hacer que la compresión se realice en el teléfono, debido a que la reducción del tráfico a través del circuito local no incrementa la capacidad del sistema.

Con la telefonía móvil hay una gran ganancia al realizar la digitalización y compresión en el teléfono, tanto que en D-AMPS tres usuarios pueden compartir un solo par de frecuencias que utilicen la multiplexión por división de tiempo. Cada par de frecuencia maneja 25 tramas/seg de 40 mseg cada uno. Además, cada trama se divide en seis ranuras de tiempo de 6.67 mseg cada una, como se muestra en la figura 2-42(a), para el par de frecuencia más bajo.

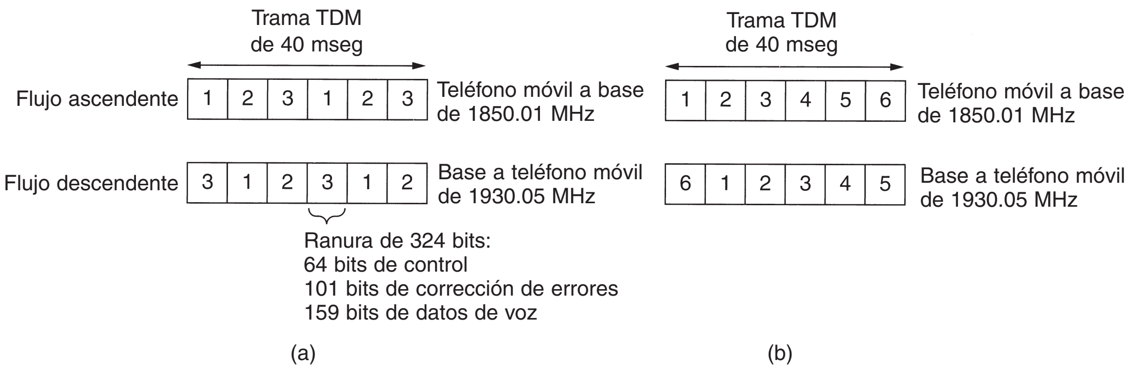


Figura 2-42. (a) Un canal D-AMPS con tres usuarios. (b) Un canal D-AMPS con seis usuarios.

Cada trama mantiene tres usuarios que se turnan para utilizar los enlaces ascendente y descendente. Por ejemplo, durante la ranura 1 de la figura 2-42(a), el usuario 1 podría transmitir a la estación base y el usuario 3 recibir de ella. Cada ranura tiene un tamaño de 324 bits de longitud, de los cuales 64 se utilizan para propósitos de protección, sincronización y control, y los 260 restantes para la carga útil del usuario. De éstos, 101 se utilizan para la corrección de errores a través del enlace de aire con ruido, por lo que a final de cuentas sólo 159 bits se dejan para la voz comprimida. Con 50 ranuras/seg, el ancho de banda disponible para la voz comprimida está por debajo de sólo 8 kbps, que es 1/7 del ancho de banda estándar PCM.

Al utilizar mejores algoritmos de compresión, es posible obtener la voz por debajo de 4 kbps, en cuyo caso seis usuarios pueden agruparse en una trama, como se ilustra en la figura 2-42(b). Desde el punto de vista del operador, poder comprimir de tres a seis veces tantos usuarios de D-AMPS en el mismo espectro que uno de AMPS es una gran ganancia y explica el porqué de la popularidad de PCS. Por supuesto, la calidad de voz a 4 kbps no se compara con lo que se podría alcanzar a 56 kbps, pero muy pocos operadores de PCS anuncian su calidad de sonido de alta fidelidad. También debe quedar claro que para datos, un canal de 8 kbps no es tan bueno como un módem antiguo de 9600 bps.

La estructura de control de D-AMPS es bastante complicada. En resumen, una supertrama está formada por grupos de 16 tramas y, algunas veces, cada supertrama tiene cierta información de control. Se utilizan seis canales principales de control: configuración del sistema, control en tiempo real y en tiempo no real, localización, respuesta de acceso y mensajes cortos. Pero de manera conceptual, funciona como AMPS. Cuando se enciende un teléfono móvil, hace contacto con la estación base para anunciarse a sí mismo y después escucha un canal de control para llamadas entrantes. Una vez que ha captado un nuevo teléfono móvil, la MTSO informa a la base doméstica del usuario dónde está, y de esta manera las llamadas se pueden enrutar en forma correcta.

Una diferencia entre AMPS y D-AMPS es la manera en que se maneja la transferencia de celdas. En AMPS, la MTSO la maneja por completo sin ayuda de los dispositivos móviles. Como se puede ver en la figura 2-42, en D-AMPS, durante 1/3 del tiempo un teléfono móvil no necesita enviar ni recibir. Utiliza estas ranuras inactivas para medir la calidad de la línea. Cuando descubre que la señal se debilita, se queja con la MTSO, la cual a continuación interrumpe la conexión, en cuyo momento el teléfono móvil trata de sintonizar una señal más fuerte desde otra estación base. Como en AMPS, le toma 300 mseg realizar la transferencia de celda. Esta técnica se conoce como **MAHO** (**Transferencia Asistida Móvil de Celda**).

GSM—Sistema Global para Comunicaciones Móviles

D-AMPS es ampliamente utilizado en Estados Unidos y (en una forma modificada) en Japón. Casi a nivel mundial, se utiliza un sistema llamado **GSM** (**Sistema Global para Comunicaciones Móviles),** e incluso se está comenzando a utilizar en Estados Unidos en una escala limitada. Para una primera aproximación, GSM es similar a D-AMPS. Los dos son sistemas celulares. En ambos se utiliza la multiplexión por división de frecuencia, en el que cada dispositivo móvil transmite en una frecuencia y recibe en una frecuencia mayor (80 MHz más arriba para D-AMPS, 55 MHz más arriba para GSM). Además, en los dos sistemas, se utiliza la multiplexión por división de tiempo para dividir un solo par de frecuencia en ranuras de tiempo compartidas por múltiples teléfonos móviles. Sin embargo, los canales GSM son mucho más anchos que los AMPS (200 kHz en comparación con 30 kHz) y almacenan relativamente pocos usuarios (8 en comparación con 3), lo que da a GSM una tasa de datos mucho más grande por usuario que D-AMPS.

A continuación describiremos brevemente algunas de las propiedades principales de GSM. Sin embargo, el estándar impreso GSM tiene cerca de 5000 páginas. Gran parte de este material se relaciona con los aspectos de ingeniería del sistema, especialmente de los receptores para

manejar la propagación de señal de múltiples trayectorias, y la sincronización de transmisores y receptores. Nada de esto se mencionará en el siguiente análisis.

Cada banda de frecuencia tiene una longitud de 200 kHz, como se muestra en la figura 2-43. Un sistema GSM tiene 124 pares de canales símplex. Cada uno de ellos tiene una longitud de 200 kHz y maneja ocho conexiones por separado, mediante la multiplexión por división de tiempo. A cada estación actualmente activa se le asigna una ranura de tiempo en el par de canal. En teoría, en cada celda se pueden manejar hasta 992 canales, aunque muchos de ellos no están disponibles, para evitar conflictos de frecuencia con las celdas vecinas. En la figura 2-43 las ocho ranuras de tiempo sombreadas pertenecen a la misma conexión, pero en cada dirección hay sólo cuatro. La transmisión y la recepción no suceden en la misma ranura de tiempo porque los radios GSM no pueden transmitir y recibir al mismo tiempo, además de que toma algo de tiempo cambiar de una a otra. Si la estación móvil a la que se le asignó 890.4/935.4 MHz y la ranura de tiempo 2 desea transmitir a la estación base, podría utilizar las cuatro ranuras de tiempo inferiores sombreadas (y las que le sigan), y colocar datos en cada ranura hasta que se hayan enviado todos los datos.

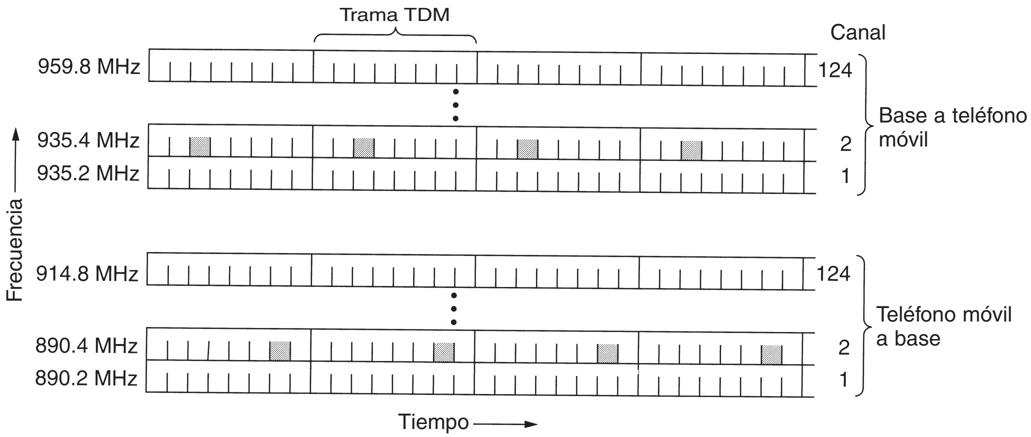


Figura 2-43. GSM utiliza 124 canales de frecuencia, cada uno de los cuales utiliza un sistema TDM de ocho ranuras.

Las ranuras TDM que se muestran en la figura 2-43 son parte de una jerarquía compleja de entramado. Cada ranura TDM tiene una estructura específica, así como grupos de ranuras TDM de multitrama, que también tienen una estructura específica. En la figura 2-44 se muestra una versión simplificada de esta jerarquía. Observe que una ranura TDM consiste en tramas de datos de 148 bits que ocupan el canal por 577 μseg (incluyendo un tiempo de protección o guarda de 30 seg después de cada ranura). Cada trama de datos inicia y termina con tres bits 0, para propósitos de delineación de trama. También contiene dos campos de *información* de 57 bits, cada uno de los cuales tiene un bit de control que indica si el siguiente campo de *información* es para voz o para datos. Entre los campos de *información* hay un campo de *sincronización* de 26 bits (entrenamiento) que es utilizado por el receptor para sincronizar los límites de la trama del emisor.

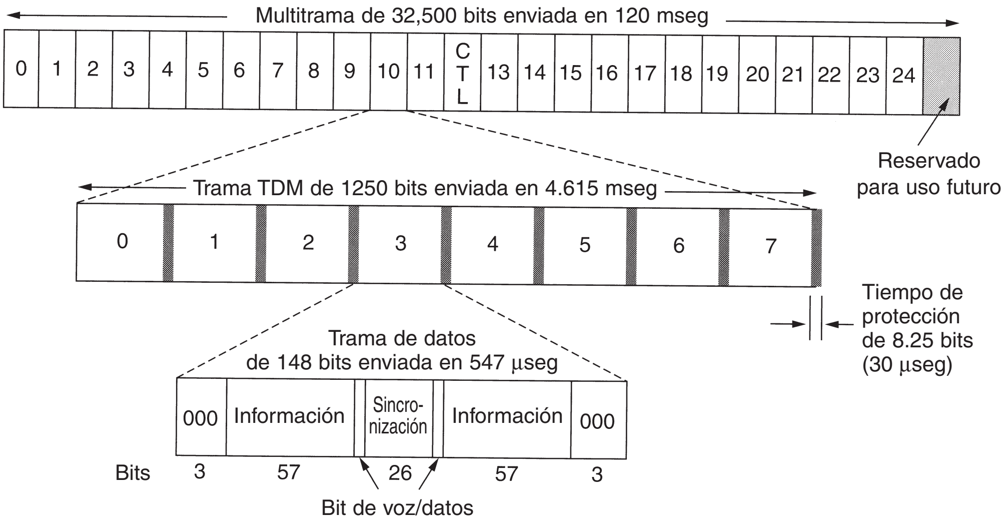


Figura 2-44. Parte de la estructura de entramado GSM.

Una trama de datos se transmite en 547 µseg, pero un transmisor sólo tiene permitido enviar una trama de datos cada 4.615 mseg, debido a que comparte el canal con otras siete estaciones. La tasa bruta de cada canal es de 270,833 bps, dividida entre ocho usuarios. Esto da un total de 33.854 kbps, más del doble que los 324 bits 50 veces por segundo de 16.2 kbps de D-AMPS. Sin embargo, al igual que con AMPS, la información adicional consume una fracción grande del ancho de banda, lo que finalmente deja 24.7 kbps de carga útil por usuario antes de la corrección de errores. Después de ésta, se dejan 13 kbps para voz, lo que da una calidad de voz sustancialmente mejor que D-AMPS (pero con el costo de utilizar de manera correspondiente más ancho de banda).

Cómo puede ver en la figura 2-44, ocho tramas de datos forman una trama TDM y 26 tramas TDM forman una multitrama de 120 mseg. De las 26 tramas TDM de una multitrama, se utiliza la ranura 12 para el control y la 25 se reserva para uso futuro, de manera que sólo 24 tramas están disponibles para el tráfico del usuario.

Sin embargo, además de la multitrama de 26 ranuras mostrado en la figura 2-44, también se utiliza una multitrama de 51 ranuras (que no se muestra). Algunas de estas ranuras se utilizan para almacenar algunos canales de control utilizados para manejar el sistema. El **canal de control de difusión** es un flujo continuo de salida de la estación base que contiene la identidad de la estación base, así como el estado del canal. Todas las estaciones móviles supervisan su fuerza de señal para ver cuándo se han movido a una nueva celda.

El **canal dedicado de control** se utiliza para actualización de localización, registro y establecimiento de llamada. En particular, cada estación base mantiene una base de datos de las estaciones móviles actualmente bajo su jurisdicción. La información necesaria para mantener esta base de datos se envía en el canal dedicado de control.

Por último, hay un **canal de control común,** que se divide en tres subcanales lógicos. El primero de estos subcanales es el **canal de localización,** que la estación base utiliza para anunciar

llamadas entrantes. Cada estación móvil lo supervisa continuamente en busca de llamadas a las que debería responder. El segundo es el **canal de acceso aleatorio,** que permite que los usuarios soliciten una ranura del canal dedicado de control. Si dos peticiones chocan, se distorsionan y se tienen que volver a realizar más tarde. La estación puede establecer una llamada utilizando la ranura del canal dedicado de control. La ranura asignada es anunciada en el tercer subcanal, el **canal de otorgamiento de acceso.**

CDMA—Acceso Múltiple por División de Código

D-AMPS y GSM son sistemas muy convencionales. Utilizan tanto FDM como TDM para dividir el espectro en canales y éstos en ranuras de tiempo. Sin embargo, hay un tercer sistema, **CDMA** (**Acceso Múltiple por División de Código**)**,** que trabaja de una forma completamente diferente. Cuando CDMA fue inicialmente propuesto, la industria tuvo casi la misma reacción que la reina Isabel cuando Colón propuso llegar a la India navegando por una ruta diferente. Sin embargo, debido a la persistencia de una compañía, Qualcomm, CDMA ha madurado al punto en el que no sólo es aceptable, sino que ahora se ve como la mejor solución técnica existente y como la base para los sistemas móviles de la tercera generación. También se utiliza ampliamente en Estados Unidos en los sistemas móviles de segunda generación, y compite de frente con D-AMPS. Por ejemplo, Sprint PCS utiliza CDMA, mientras que AT&T Wireless utiliza D-AMPS. CDMA se describe en el International Standard IS-95 y algunas veces se hace referencia a él mediante ese nombre. También se utiliza el nombre **cdmaOne.**

CDMA es completamente diferente de AMPS, D-AMPS y GSM. En lugar de dividir el rango de frecuencia permitida en algunos cientos de canales estrechos, CDMA permite que cada estación transmita todo el tiempo a través de todo el espectro de frecuencia. Se utiliza la teoría de codificación para separar múltiples transmisiones simultáneas. CDMA no supone que las tramas que colisionan son totalmente distorsionadas. En su lugar, asume que se agregan múltiples señales en forma lineal.

Antes de adentrarnos en el algoritmo, consideremos una analogía: una sala de espera de un aeropuerto con muchas parejas de personas conversando. TDM se compara con todas las personas que están en medio de la sala pero que esperan su turno para hablar. FDM se compara con las personas que están en grupos separados ampliamente, y cada grupo tiene su propia conversación al mismo tiempo, aunque de manera independiente, que los otros. CDMA se compara con el hecho de que todas las personas estén en medio de la sala hablando al mismo tiempo, pero cada pareja hablando en un lenguaje diferente. La pareja que habla francés se concentra en el francés, rechazando todo lo que no sea francés como si fuera ruido. Por lo tanto, la clave de CDMA es tener la capacidad de extraer la señal deseada y rechazar todo lo demás como ruido aleatorio. A continuación se da una descripción algo simplificada de CDMA.

En CDMA, cada tiempo de bit se subdivide en *m* intervalos cortos llamados **chips.** Por lo general, hay 64 o 128 chips por bit, pero en el ejemplo que se da a continuación por simplicidad utilizaremos 8 chips/bit.

A cada estación se le asigna un código único de *m* bits llamado **secuencia de chip.** Para transmitir un bit 1, una estación envía su secuencia de chips. Para transmitir un bit 0, envía el comple-

mento de uno de su secuencia de chips. No se permiten otros patrones. Por lo tanto, para $m = 8$, si a la estación *A* se le asigna la secuencia de chips 00011011, envía un bit 1 mediante el envío de 00011011 y un bit 0 mediante el envío de 11100100.

El incremento de la cantidad de información que se va a enviar de *b* bits/seg a *mb* chips/seg sólo puede realizarse si el ancho de banda disponible se incrementa por un factor de *m*, lo que hace de CDMA una forma de comunicaciones de espectro disperso (suponiendo que no haya cambios en las técnicas de codificación o modulación). Si tenemos una banda de 1 MHz disponible para 100 estaciones, con FDM cada una tendría 10 kHz y podría enviarse a 10 kbps (suponiendo 1 bit por Hz). Con CDMA, cada estación utiliza completamente el megahertzio, por lo que la tasa de chips es de 1 megachip por segundo. Con menos de 100 chips por bit, el ancho de banda efectivo por estación es mayor para CDMA que FDM, y el problema de asignación de canal se resuelve.

Para propósitos de enseñanza, es más conveniente utilizar una notación bipolar, donde el 0 binario es −1 y el 1 es +1. Mostraremos las secuencias de chips entre paréntesis, de manera que 1 bit para la estación *A* ahora se vuelve (−1−1−1+1+1−1+1+1). En la figura 2-45(a) mostramos las secuencias de chips asignadas a cuatro estaciones de ejemplo. En la figura 2-45(b) las mostramos en nuestra notación bipolar.

(a)	(b)
A: 0 0 0 1 1 0 1 1	A:(−1−1−1+1+1−1+1+1)
B: 0 0 1 0 1 1 1 0	B:(−1−1+1−1+1+1+1−1)
C: 0 1 0 1 1 1 0 0	C:(−1+1−1+1+1+1−1−1)
D: 0 1 0 0 0 0 1 0	D:(−1+1−1−1−1−1+1−1)

Seis ejemplos:

– –1–	**C**	S_1 = (−1+1−1+1+1+1−1−1)
–11–	**B** + **C**	S_2 = (−2 0 0 0+2+2 0−2)
10– –	**A** + **B**	S_3 = (0 0−2+2 0−2 0+2)
101–	**A** + **B** + **C**	S_4 = (−1+1−3+3+1−1−1+1)
1111	**A** + **B** + **C** + **D**	S_5 = (−4 0−2 0+2 0+2−2)
1101	**A** + **B** + $\overline{\mathbf{C}}$ + **D**	S_6 = (−2−2 0−2 0−2+4 0)

(c)

S_1 • C = (1+1+1+1+1+1+1+1)/8 = 1
S_2 • C = (2+0+0+0+2+2+0+2)/8 = 1
S_3 • C = (0+0−2+2+0−2+0−2)/8 = 0
S_4 • C = (1+1+3+3+1−1+1−1)/8 = 1
S_5 • C = (4+0+2+0+2+0−2+2)/8 = 1
S_6 • C = (2−2+0−2+0−2−4+0)/8 = −1

(d)

Figura 2-45. (a) Secuencias de chips binarios para cuatro estaciones. (b) Secuencias de chips bipolares. (c) Seis ejemplos de transmisiones. (d) Recuperación de la señal s de la estación C.

Cada estación tiene su propia y única secuencia de bits. Utilicemos el símbolo **S** para indicar el vector de *m* chips para la estación *S*, y **S** para su negación. Todas las secuencias de chips son

ortogonales y apareadas, lo cual quiere decir que el producto interno normalizado de cualquiera de dos secuencias distintas de chips, **S** y **T** (escritas como **S•T**), es 0. Tales secuencias ortogonales de chips se pueden generar utilizando un método conocido como **código de Walsh.** En términos matemáticos, la ortogonalidad de las secuencias de chips se puede expresar como se muestra a continuación:

$$\mathbf{S}\bullet\mathbf{T} \equiv \frac{1}{m}\sum_{i=1}^{m} S_i T_i = 0 \qquad (2\text{-}4)$$

En términos simples, entre más parecidos sean los pares, más diferentes serán. Esta propiedad de ortogonalidad será crucial más adelante. Observe que si **S•T** = 0, entonces $\mathbf{S}\bullet\overline{\mathbf{T}}$ también es 0. El producto interno normalizado de cualquier secuencia de chips por sí misma es 1:

$$\mathbf{S}\bullet\mathbf{S} = \frac{1}{m}\sum_{i=1}^{m} S_i S_i = \frac{1}{m}\sum_{i=1}^{m} S_i^2 = \frac{1}{m}\sum_{i=1}^{m} (\pm 1)^2 = 1$$

Esto continúa porque cada uno de los términos *m* del producto interno es 1, por lo que la suma es *m*. Además, observe que $\mathbf{S}\bullet\overline{\mathbf{S}} = -1$.

Durante cada tiempo de bit, una estación puede transmitir un 1 enviando su secuencia de chips, puede transmitir un 0 enviando el negativo de su secuencia de chips, o puede quedarse en silencio y no transmitir nada. Por el momento, asumimos que todas las estaciones están sincronizadas, por lo que todas las secuencias de chips comienzan al mismo tiempo.

Cuando dos o más estaciones transmiten de manera simultánea, sus señales bipolares se agregan de manera lineal. Por ejemplo, si en un periodo de chips tres estaciones envían +1 y una estación envía −1, el resultado es +2. Uno podría pensar que esto es como agregar voltaje: tres estaciones enviando +1 voltio y una estación enviando −1 voltio da un total de 2 voltios.

En la figura 2-45(c) vemos seis ejemplos de una o más estaciones que transmiten al mismo tiempo. En el primer ejemplo, *C* transmite un bit 1, por lo que simplemente obtenemos la secuencia de chips de *C*. En el segundo ejemplo, tanto *B* como *C* transmiten bits 1, por lo que obtenemos la suma de sus secuencias de chips bipolares, principalmente:

$$(-1-1+1-1+1+1+1-1) + (-1+1-1+1+1+1-1-1) = (-2\ 0\ 0\ 0{+}2{+}2\ 0{-}2)$$

En el tercer ejemplo, la estación *A* envía un 1 y la estación *B* envía un 0. Las demás están calladas. En el cuarto ejemplo, *A* y *C* envían un 1 mientras que *B* envía un bit 0. En el quinto ejemplo, todas las estaciones envían un bit 1. Finalmente, en el último ejemplo, *A, B* y *D* envían un bit 1, mientras que *C* envía un bit 0. Observe que cada una de las secuencias S_1 a S_6 que se dan en la figura 2-45(c) sólo representa un tiempo de bit.

Para recuperar el flujo de bits de una estación individual, el receptor debe conocer con anticipación la secuencia de chips de esa estación. Realiza la recuperación calculando el producto interno normalizado de la secuencia de chips recibida (la suma lineal de todas las estaciones que transmitieron) y la secuencia de chips de la estación cuyo flujo de bits se está tratando de recuperar. Si la secuencia de chips recibida es **S** y el receptor está tratando de escuchar en una estación cuya secuencia de chips es **C,** simplemente calcula el producto interno normalizado, **S•C.**

Para ver por qué funciona esto, simplemente imagine que dos estaciones, *A* y *C*, transmiten un bit 1 al mismo tiempo que *B* transmite un bit 0. El receptor ve la suma $\mathbf{S} = \mathbf{A} + \overline{\mathbf{B}} + \mathbf{C}$ y calcula

$$\mathbf{S} \bullet \mathbf{C} = (\mathbf{A} + \overline{\mathbf{B}} + \mathbf{C}) \bullet \mathbf{C} = \mathbf{A} \bullet \mathbf{C} + \overline{\mathbf{B}} \bullet \mathbf{C} + \mathbf{C} \bullet \mathbf{C} = 0 + 0 + 1 = 1$$

Los primeros dos términos desaparecen porque todos los pares de secuencias de chips han sido cuidadosamente elegidas para ser ortogonales, como se muestra en la ecuación 2-4. Ahora ya debería ser claro por qué esta propiedad debe imponerse en las secuencias de bits.

Un punto de vista alterno acerca de esta situación es imaginar que las tres secuencias vienen por separado, en lugar de sumadas. Después, el receptor podría calcular el producto interno con cada uno por separado y sumar los resultados. Debido a la propiedad ortogonal, todos los productos internos, excepto **C•C** serían 0. Sumar los productos internos y luego calcularlos es lo mismo que calcularlos y luego sumarlos.

Para hacer más concreto el proceso de decodificación, consideremos nuevamente los seis ejemplos de la figura 2-45(c), como se ilustra en la figura 2-45(d). Suponga que el receptor está interesado en extraer el bit enviado por la estación *C* de cada una de las seis sumas S_1 a S_6. Calcula el bit sumando los productos apareados de la **S** recibida y del vector **C** de la figura 2-45(b) y, después, tomando 1/8 del resultado (debido a que aquí $m = 8$). Como se muestra, el bit correcto es decodificado cada vez. Es como hablar francés.

En un sistema CDMA ideal y libre de ruido, es posible hacer que la capacidad (es decir, el número de estaciones) sea arbitrariamente grande, de la misma manera en que lacapacidad de un canal Nyquist sin ruido puede hacerse grande de manera arbitraria utilizando más y más bits por muestra. En la práctica, las limitaciones físicas reducen la capacidad en forma considerable. Primero, hemos asumido que todos los chips están sincronizados. En la realidad, tal sincronización es imposible. Lo que puede hacerse es que el emisor y el receptor se sincronicen haciendo que el emisor transmita una secuencia predefinida de chips que sea lo suficientemente grande para que el receptor pueda detectarla. En consecuencia, todas las demás transmisiones (sin sincronizar) se consideran ruido aleatorio. Sin embargo, si no hay muchas de ellas, el algoritmo básico de decodificación aún funciona bien. Existe bastante literatura que describe la superposición de las secuencias de chips a nivel de ruido (Pickholtz y cols., 1982). Como se podría esperar, entre mayor sea la secuencia de chips, mayor será la probabilidad de detectarla de manera correcta en caso de que haya ruido. Para una mayor confiabilidad, la secuencia de bits puede usar un código de corrección de errores. Las secuencias de chips nunca utilizan códigos de corrección de errores.

Una suposición implícita en nuestro análisis es que los niveles de potencia de todas las estaciones son los mismos que el receptor captó. Por lo general, CDMA se utiliza para sistemas inalámbricos con una estación base fija y muchas estaciones móviles a distancias distintas de ella. Los niveles de potencia recibidos en la estación base dependen de qué tan lejos estén los transmisores. Una buena heurística aquí es que cada estación móvil transmita a la estación base con una magnitud de potencia inversa a la que recibe de la estación base. En otras palabras, una estación móvil que reciba una señal débil de la estación base utilizará más potencia que una que reciba una señal fuerte. La estación base también puede dar órdenes explícitas a las estaciones móviles para incrementar o decrementar su potencia de transmisión.

También hemos dado por hecho que el receptor sabe quién es el emisor. Al principio, contando con suficiente capacidad, el receptor puede escuchar al mismo tiempo a todos los emisores mediante la ejecución paralela del algoritmo de decodificación para cada uno de ellos. En la realidad, esto es más fácil de decir que de hacer. CDMA también tiene otros aspectos complicados que se han pasado por alto en esta breve introducción. A pesar de todo, CDMA es un esquema brillante y se está introduciendo rápidamente en la comunicación inalámbrica móvil. Por lo general, funciona en una banda de 1.25 MHz (en comparación con los 30 kHz de D-AMPS y los 200 kHz de GSM), pero maneja muchos más usuarios en esa banda que cualquiera de los otros sistemas. En la práctica, el ancho de banda disponible para cada usuario es tan bueno como el disponible en GSM y, con frecuencia, mejor.

Para un entendimiento muy profundo de CDMA, vea (Lee y Miller, 1998). Un esquema disperso alterno, en donde la dispersión se realiza a través del tiempo en lugar de la frecuencia, se describe en (Crespo y cols., 1995). Otro esquema más se describe en (Sari y cols., 2000). Todas estas referencias requieren conocimientos de ingeniería de comunicaciones.

2.6.3 Teléfonos móviles de tercera generación: voz y datos digitales

¿Cuál es el futuro de la telefonía móvil? Echemos un vistazo. Hay algunos factores que están impulsando a la industria. Primero, el tráfico de datos ya excede el tráfico de voz en la red fija y está creciendo de manera exponencial, mientras que el tráfico de voz es en esencia plano. Muchos expertos de la industria esperan que muy pronto el tráfico de datos domine la voz en dispositivos móviles. Segundo, las industrias telefónica, de entretenimiento y de cómputo han adoptado formatos digitales y están convergiendo rápidamente. Muchas personas están deseosas de un dispositivo portable y ligero que actúe como teléfono, reproductor de CDs, reproductor de DVDs, terminal de correo electrónico, interfaz para Web, máquina de juegos, procesador de texto, etcétera, todo con conectividad inalámbrica a Internet en todo el mundo con un ancho de banda alto. Este dispositivo y cómo conectarlo es de lo que trata la telefonía móvil de tercera generación. Para mayor información, vea (Huber y cols., 2000, y Sarikaya, 2000).

En 1992, la ITU trató de llevar a cabo este sueño y creó un diseño para alcanzarlo, llamado **IMT-2000.** IMT son las siglas de **Telecomunicaciones Móviles Internacionales.** Se le agregó el número 2000 por tres razones: (1) era el año en que se suponía debería funcionar, (2) era a la frecuencia a la que se suponía que trabajaría (en MHz) y (3) era el ancho de banda que el servicio debería tener (en kHz).

No cumplió con nada de lo anterior. En el 2000 no se implementó nada. La ITU recomendó que todos los gobiernos reservaran espectro de 2 GHz a fin de que los dispositivos pudieran llevarse a cualquier país y funcionaran a la perfección. China reservó el ancho de banda requerido pero nadie más lo hizo. Por último, se admitió que 2 Mbps no son factibles para usuarios que se desplazan *mucho* (debido a la dificultad de realizar transferencias de celdas con la rapidez necesaria). Los 2 Mbps son más factibles para usuarios fijos (lo cual podrá competir con ADSL), 384 kbps para usuarios a pie y 144 kbps para conexiones en automóviles. Sin embargo, el área completa de **3G,** como se ha llamado, es un gran caldero de actividad. La tercera generación podría ser menor y llegar más tarde de lo que originalmente se esperaba, pero seguramente sucederá.

Los servicios básicos que se supone que la red IMT-2000 proporcionará a sus usuarios son:

1. Transmisión de voz de alta calidad.
2. Mensajería (lo cual reemplazará al correo electrónico, a los faxes, a SMS, a los salones de conversación, etcétera).
3. Multimedia (reproducir música, ver vídeos, películas, televisión, etcétera).
4. Acceso a Internet (navegar por Web, incluyendo páginas con audio y vídeo).

Otros servicios adicionales podrían ser la videoconferencia, la telepresencia, los juegos en grupo y el comercio móvil (pasar su teléfono por el cajero para pagar en una tienda). Además, se supone que todos estos servicios estén disponibles a nivel mundial (con conexión automática vía satélite cuando no se encuentre una red terrestre), de manera instantánea (siempre conectado) y con garantía de calidad de servicio.

La ITU visualizó una tecnología sencilla a nivel mundial para IMT-2000, de manera que los fabricantes pudieran construir un solo dispositivo que pudiera venderse y utilizarse en cualquier parte del mundo (similar a un reproductor de CDs y las computadoras, y diferente de los teléfonos y televisiones móviles). Tener una sola tecnología también podría facilitar la vida de los operadores de red y alentaría a más personas a utilizar los servicios. Las guerras de formato, como la batalla entre Beta y VHS cuando aparecieron por primera vez las videograbadoras, no son buenas para los negocios.

Se realizaron varias propuestas y, después de varias selecciones, aparecieron las dos principales. La primera, **W-CDMA (CDMA de Banda Ancha),** fue propuesta por Ericsson. Este sistema utiliza espectro disperso de secuencia directa del tipo que describimos anteriormente. Se ejecuta en una banda ancha de 5 MHz y se ha diseñado para interactuar con redes GSM aunque no tiene compatibilidad hacia atrás con GSM. Sin embargo, tiene la propiedad de que el invocador puede salir de una celda W-CDMA y entrar a una celda GSM sin perder la llamada. Este sistema fue impulsado por la Unión Europea, que lo llamó **UMTS (Sistema Universal de Telecomunicaciones Móviles).**

El otro contendiente era **CDMA2000,** propuesto por Qualcomm. Éste también es un diseño de espectro disperso de secuencia directa, básicamente una extensión de IS-95 y es compatible hacia atrás con él. También utiliza un ancho de banda de 5-MHz, pero no ha sido diseñado para interactuar con GSM y no puede entregar llamadas a una celda GSM (ni a una celda DAMPS). Algunas de las diferencias técnicas con respecto a W-CDMA son las siguientes: una tasa de chips diferente, un tiempo de trama diferente, se utiliza un espectro diferente y la sincronización de tiempo se realiza de una manera diferente.

Si los ingenieros de Ericsson y de Qualcomm se reunieran en un cuarto y se les pidiera que crearan un diseño común, tal vez lo podrían hacer. Después de todo, el principio detrás de ambos sistemas es CDMA en un canal de 5 MHz y nadie está dispuesto a morir por esta tasa de chips elegida. El problema real no es la ingeniería, sino las políticas (lo usual). Europa quería un sistema que pudiera interactuar con GSM; Estados Unidos quería un sistema que fuera compatible con uno que ya se distribuía ampliamente en Estados Unidos (IS-95). Cada lado también

mantenía su compañía local (Ericsson se encuentra en Suecia; Qualcomm en California). Por último, Ericsson y Qualcomm se involucraron en numerosas demandas por sus respectivas patentes de CDMA.

En marzo de 1999, las dos compañías resolvieron los problemas legales cuando Ericsson estuvo de acuerdo en comprar la infraestructura de Qualcomm. También estaban de acuerdo en un estándar sencillo 3G, con múltiples opciones incompatibles, que en gran parte simplemente disfraza las diferencias técnicas. A pesar de estas disputas, los dispositivos y servicios de 3G probablemente comiencen a aparecer en los años venideros.

Se ha escrito mucho acerca de los sistemas 3G, la mayor parte los describen como lo mejor desde el pan en rebanadas. Algunas referencias son (Collins y Smith, 2001; De Vriendt y cols., 2002; Harte y cols., 2002; Lu, 2002, y Sarikaya, 2000). Sin embargo, algunos disidentes piensan que la industria está apuntando en la dirección equivocada (Garber, 2002, y Goodman, 2000).

Mientras se espera que termine la batalla por 3G, algunos operadores están dando cautelosamente un pequeño paso en dirección a 3G al ir a lo que algunas veces se llama **2.5G,** aunque 2.1G sería más preciso. Tal sistema es **EDGE** (**Tasa de Datos Mejorada para la Evolución del GSM**)**,** que simplemente es GSM con más bits por baudio. El problema es que más bits por baudio también significan más errores por baudio, por lo que EDGE tiene nueve esquemas diferentes para modulación y corrección de errores, que difieren en la cantidad de ancho de banda que se dedica a arreglar los errores introducidos por la velocidad más alta.

Otro esquema de 2.5G es **GPRS** (**Servicio de Radio de Paquetes Generales**)**,** que es una red de paquetes superpuestos encima de D-AMPS o GSM. Permite que las estaciones móviles envíen y reciban paquetes IP en una celda que se ejecuta en un sistema de voz. Cuando GPRS está en operación, algunas ranuras de tiempo en algunas frecuencias se reservan para el tráfico de paquetes. La estación base puede manejar de manera dinámica el número y la ubicación de las ranuras de tiempo, dependiendo de la tasa de voz sobre el tráfico de datos de la celda.

Las ranuras de tiempo disponibles se dividen en varios canales lógicos, utilizados para propósitos diferentes. La estación base determina qué canales lógicos se asignan en qué ranuras de tiempo. Un canal lógico se utiliza para bajar paquetes de la estación base a algunas estaciones móviles, y cada paquete indica a quién va destinado. Para enviar un paquete IP, una estación móvil solicita una o más ranuras de tiempo enviando una petición a la estación base. Si la petición llega sin daño alguno, la estación base anuncia la frecuencia y las ranuras de tiempo asignadas al móvil para enviar el paquete. Una vez que el paquete llega a la estación base, se transfiere a Internet mediante una conexión de cable. Puesto que GPRS sólo es una superposición en el sistema de voz existente, es en el mejor de los casos una medida provisional hasta que llegue 3G.

Aunque las redes 3G aún no se han distribuido ampliamente, algunos investigadores consideran a 3G como un asunto terminado y ya no están interesados. Estas personas ahora trabajan en sistemas 4G (Berezdivin y cols., 2002; Guo y Chaskar, 2002; Huang y Zhuang, 2002; Kellerer y cols., 2002, y Misra y cols., 2002). Entre las características propuestas de los sistemas 4G se encuentran un ancho de banda alto, ubicuidad (conectividad en cualquier lado), integración perfecta con redes de cable y especialmente IP, manejo de espectro y de recursos adaptable, radios de software y alta calidad de servicio para multimedia.

Por otro lado, se están estableciendo tantos puntos de acceso a LANs inalámbricas 802.11 por todos lados, que algunas personas piensan que 3G no solamente no es un asunto terminado, sino que está condenado al fracaso. Bajo esta óptica, las personas irán de un punto de acceso 802.11 a otro para mantenerse conectados. Decir que la industria está en medio de un intenso proceso de cambio es poco. Echemos un vistazo en cinco años para ver qué pasa.

2.7 TELEVISIÓN POR CABLE

Hemos estudiado tanto los sistemas inalámbricos como los fijos con suficiente detalle. Ambos jugarán un papel importante en las redes futuras. Sin embargo, hay una alternativa para la conectividad de redes fija que está tomando mucha importancia: las redes de televisión por cable. Muchas personas ya tienen su teléfono y servicio de Internet a través de cable, y los operadores de cable están trabajando arduamente para incrementar su participación de mercado. En las siguientes secciones analizaremos con detalle a la televisión por cable como un sistema de conectividad de redes y lo compararemos con los sistemas telefónicos que acabamos de estudiar. Para mayor información sobre el cable, vea (Laubach y cols., 2001; Louis, 2002; Ovadia, 2001, y Smith, 2002).

2.7.1 Televisión por antena comunal

La televisión por cable se concibió en la última parte de la década de 1940 como una forma de proporcionar mejor recepción a las personas que viven en las áreas rurales o montañosas. El sistema consistió inicialmente en una antena grande en la cima de una colina para captar la señal de televisión, un amplificador, llamado **amplificador *head end*,** para reforzarla y un cable coaxial para enviarla a las casas de las personas, como se ilustra en la figura 2-46.

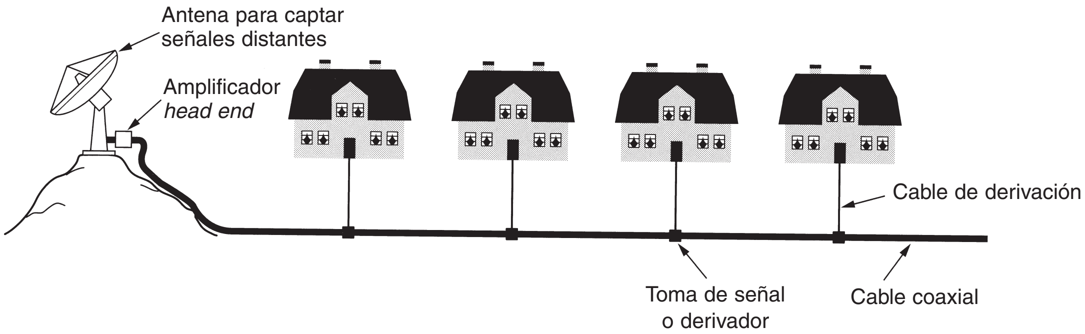


Figura 2-46. Sistema de televisión por cable antiguo.

En sus primeros años, la televisión por cable fue llamada **televisión por antena comunal.** Era un negocio familiar; cualquiera que fuera hábil con la electrónica podía establecer un servicio para su comunidad, y los usuarios podían pagarlo en conjunto. Conforme el número de suscriptores crecía, se unían cables adicionales al cable original y se agregaban amplificadores. La transmisión era de una vía, del amplificador *head end* a los usuarios. En 1970 ya existían miles de sistemas independientes.

En 1974, Time, Inc. inició un nuevo canal, Home Box Office, con contenido nuevo (películas) y que se distribuía sólo por cable. Le siguieron otros canales que se transmitían sólo por cable y cuyo contenido eran noticias, deportes, cocina entre muchos otros. Este desarrollo dio origen a dos cambios en la industria. Primero, las grandes compañías comenzaron a comprar sistemas de cable existentes e instalar nuevo cable para adquirir más suscriptores. Segundo, surgió la necesidad de conectar múltiples sistemas, por lo general en ciudades distantes, para distribuir los nuevos canales por cable. Las compañías de cable comenzaron a instalar cable entre ciudades para conectarlas en un solo sistema. Este patrón fue similar a lo que pasó en la industria telefónica 80 años antes con la conexión de las oficinas centrales locales previamente aisladas para hacer posible las llamadas de larga distancia.

2.7.2 Internet a través de cable

A través de los años, el sistema de televisión por cable creció y los cables entre las distintas ciudades se reemplazaron por fibra de ancho de banda alto, de manera similar a lo que sucedió con el sistema telefónico. Un sistema con fibra para distancias considerables y cable coaxial para las casas se conoce como sistema **HFC** (**Red Híbrida de Fibra Óptica y Cable Coaxial**). Los convertidores electroópticos que interactúan entre las partes óptica y eléctrica del sistema se llaman **nodos de fibra.** Debido a que el ancho de banda de la fibra es mucho mayor al del cable coaxial, un nodo de fibra puede alimentar múltiples cables coaxiales. En la figura 2-47(a) se muestra parte de un sistema moderno HFC.

En los años recientes, muchos operadores de cable han decidido entrar al negocio de acceso a Internet y con frecuencia también al de la telefonía. Sin embargo, las diferencias técnicas entre la planta de cable y la de telefonía tiene mucho que ver con respecto a lo que se tiene que hacer para alcanzar esas metas. Por un lado, todos los amplificadores de una vía del sistema tienen que reemplazarse por amplificadores de dos vías.

Sin embargo, hay otra diferencia entre el sistema HFC de la figura 2-47(a) y el sistema telefónico de la figura 2-47(b) que es más difícil eliminar. En los vecindarios, muchas casas comparten un solo cable, mientras que en el sistema telefónico, cada casa tiene su propio circuito local privado. Cuando se emplea en la difusión de televisión, esta compartición no tiene importancia. Todos los programas se difunden a través del cable y no importa si hay diez o tres o 10,000 televidentes. Cuando el mismo cable se utiliza para el acceso a Internet, el hecho de que haya 10 o 10,000 usuarios tiene mucha importancia. Si un usuario decide descargar un archivo muy grande, ese ancho de banda se les resta a otros usuarios. Entre más usuarios haya, habrá más competencia por el ancho de banda. El sistema telefónico no tiene esta propiedad particular: descargar un

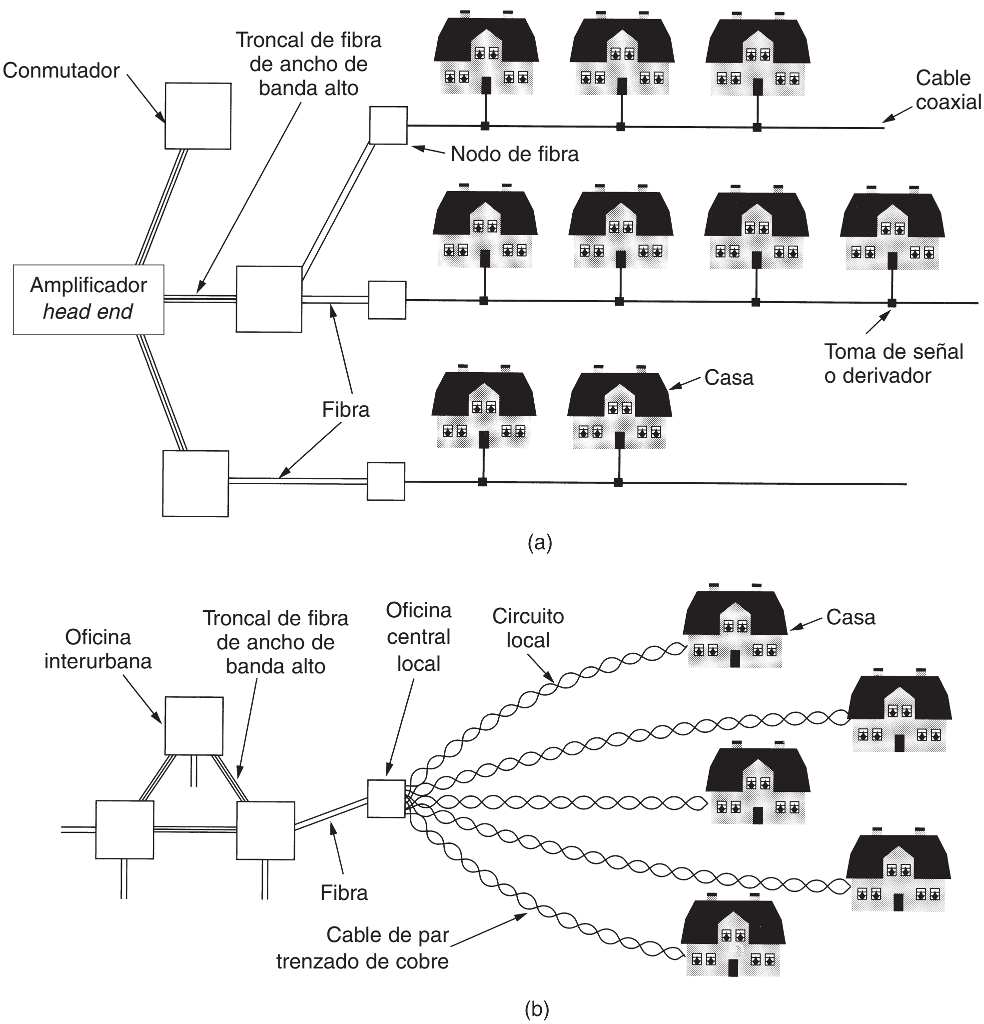


Figura 2-47. (a) Televisión por cable. (b) El sistema telefónico fijo.

archivo grande a través de una línea ADSL no reduce el ancho de banda del vecino. Por otra parte, el ancho de banda del cable coaxial es mucho mayor que el del cable de par trenzado.

La forma en que la industria del cable ha abordado este problema es dividiendo los cables largos y conectándolos directamente al nodo de fibra. El ancho de banda que el amplificador *head end* proporciona a cada nodo de fibra es infinito; por lo tanto, siempre y cuando no haya demasiados suscriptores en cada segmento del cable, la cantidad de tráfico será manejable. En la actualidad, los cables típicos tienen de 500–2000 casas, pero entre más y más gente se suscribe a Internet a través de cable, la carga podría volverse demasiada, lo que requeriría más divisiones y más nodos de fibra.

2.7.3 Asignación de espectro

Deshacerse de todos los canales de TV y utilizar la infraestructura de cable tan sólo para el acceso a Internet tal vez generaría una cantidad considerable de clientes iracundos, por lo que las compañías de cable dudan en hacer esto. Además, la mayoría de las ciudades regulan estrictamente lo que hay en el cable, por lo que podría no permitirse que los operadores de cable hagan esto aunque realmente deseen hacerlo. Como consecuencia, necesitan encontrar una manera de que las televisiones e Internet coexistan en el mismo cable.

Los canales de televisión por cable en Norteamérica normalmente ocupan la región de 54–550 MHz (excepto por la radio FM de 88 a 108 MHz). Estos canales tienen 6 MHz de ancho, incluyendo las bandas de protección. En Europa el extremo inferior por lo general es de 65 MHz y los canales tienen un ancho de 6–8 MHz para la resolución más alta requerida por PAL y SECAM, pero en lo demás el esquema de asignación es similar. La parte baja de la banda no se utiliza. Los cables modernos también pueden operar bien arriba de 550 MHz, con frecuencia a 750 MHz o más. La solución elegida fue introducir canales ascendentes en la banda de 5–42 MHz (un poco más arriba en Europa) y utilizar las frecuencias en el extremo superior para el flujo descendente. El espectro del cable se ilustra en la figura 2-48.

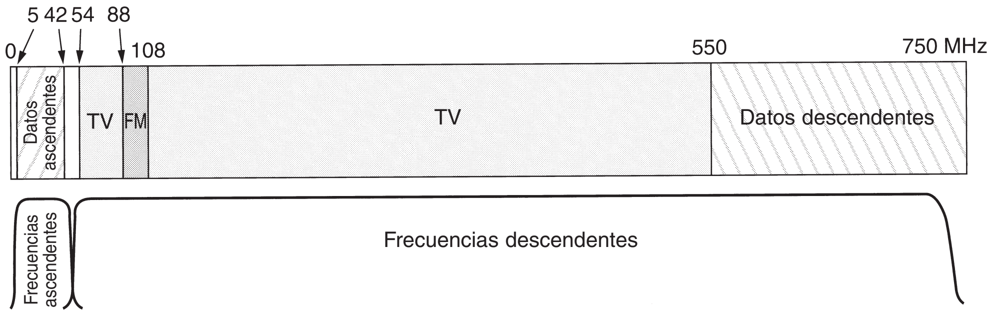


Figura 2-48. Asignación de frecuencia en un sistema típico de cable por TV utilizado para acceso a Internet.

Observe que debido a que todas las señales de televisión son descendentes, es posible utilizar amplificadores ascendentes que funcionen en la región de 5–42 MHz y amplificadores descendentes que sólo funcionen a 54 MHz y mayor, como se muestra en la figura. Por lo tanto, obtenemos una asimetría en los anchos de banda de los flujos ascendente y descendente debido a que hay disponible más espectro arriba del espectro de la televisión que abajo. Por otra parte, la mayor parte del tráfico probablemente sea hacia abajo, por lo que los operadores de cable no están descontentos con este hecho. Como vimos anteriormente, las compañías telefónicas por lo general ofrecen un servicio DSL asimétrico, aunque no tienen ninguna razón técnica para hacerlo.

Los cables coaxiales largos no son mejores para transmitir señales digitales que los circuitos locales largos, por lo que aquí también se necesita la modulación analógica. El esquema usual es tomar cada canal descendente de 6 u 8 MHz y modularlo con QAM-64 o, si la calidad del cable

es muy buena, QAM-256. Con un canal de 6 MHz y QAM-64, obtenemos casi 36 Mbps. Cuando se sustrae la sobrecarga, la carga útil de la red es de aproximadamente 27 Mbps. Con QAM-256, dicha carga es de aproximadamente de 39 Mbps. Los valores europeos son 1/3 más grandes.

Para el flujo ascendente, incluso QAM-64 no funciona bien. Hay mucho ruido proveniente de las microondas, radios CB y otras fuentes, y por esto se utiliza un esquema más conservador —QPSK. Este método (mostrado en la figura 2-25) proporciona 2 bits por baudio en lugar de los 6 u 8 bits que QAM proporciona en los canales descendentes. En consecuencia, la asimetría entre el ancho de banda ascendente y el descendente es mayor de la que se sugiere en la figura 2-48.

Además de actualizar los amplificadores, el operador también tiene que actualizar el amplificador *head end*, de un amplificador tonto a un sistema de cómputo digital inteligente con una interfaz de fibra de ancho de banda alto a un ISP. Por lo general, el nombre también se actualiza, de "amplificador *head end*" a **CMTS** (**Sistema de Terminación de Módem de Cable**)**.** En el siguiente texto, nos referiremos a él como "amplificador *head end*".

2.7.4 Módems de cable

El acceso a Internet requiere un módem de cable, un dispositivo que tiene dos interfaces: una en la computadora y la otra en la red de cable. En los primeros años de Internet por cable, cada operador tenía un módem de cable patentado, que era instalado por un técnico de la compañía de cable. Sin embargo, pronto quedó claro que un estándar abierto podría crear un mercado de módems de cable competitivo y bajar los precios, con lo que se alentaría el uso del servicio. Además, al permitir que los clientes compren los módems de cable en tiendas y que los instalen ellos mismos (como lo hicieron con los módems de teléfono V.9x) se podrían eliminar los temidos camiones de la compañía de cable.

En consecuencia, los operadores de cable más grandes se unieron a una compañía llamada CableLabs para producir un módem de cable estándar y probar la compatibilidad de productos. Este estándar, llamado **DOCSIS** (**Especificación de Interfaz para Servicio de Datos por Cable**)**,** está comenzando a reemplazar a los módems patentados. La versión europea se llama **EuroDOCSIS.** Sin embargo, no a todos los operadores de cable les gusta la idea de un estándar, debido a que muchos de ellos estaban ganando bastante dinero rentando sus módems a sus clientes cautivos. Un estándar abierto con docenas de fabricantes vendiendo módems de cable en tiendas termina con esta práctica tan lucrativa.

La interfaz módem a computadora es directa. En la actualidad, con frecuencia es la Ethernet a 10-Mbps (y en ocasiones es USB). En el futuro, todo el módem podría ser una pequeña tarjeta conectada en la computadora, al igual que con los módems internos V.9x.

El otro extremo es más complicado. Una gran parte del estándar tiene que ver con la ingeniería de radio, tema que está muy alejado del objetivo de este libro. La única parte que vale la pena mencionar aquí es que los módems de cable, al igual que los ADSL, siempre están activos. Establecen una conexión cuando se encienden y la mantienen todo el tiempo que tengan energía, debido a que los operadores de cable no cobran por el tiempo de conexión.

Para entender mejor cómo funcionan, veamos lo que pasa cuando un módem de cable se conecta y activa. El módem explora los canales descendentes en busca de un paquete especial que el

amplificador *head end* transmite periódicamente para proporcionar parámetros del sistema a los módems que se acaban de conectar. Al encontrar este paquete, el nuevo módem anuncia su presencia en uno de los canales ascendentes. El amplificador *head end* responde asignando al módem a sus canales ascendente y descendente. Estas asignaciones pueden cambiarse más tarde si el amplificador *head end* estima que es necesario balancear la carga.

A continuación, el módem determina su distancia con respecto al amplificador *head end* enviándole un paquete especial y tomando el tiempo que tarda en llegar la respuesta. Este proceso se conoce como **alineación** (*ranging*). Es importante que el módem sepa su distancia para reubicar el camino por el que los canales ascendentes funcionan y para obtener la temporización correcta. Dichos canales se dividen en **minirranuras.** Cada paquete ascendente debe ajustarse en una o más minirranuras consecutivas. El amplificador *head end* indica en forma periódica el inicio de una nueva ronda de minirranuras, pero la señal de partida no es escuchada en todos los módems de manera simultánea debido al tiempo de propagación en el cable. Al saber la distancia que separa al módem del amplificador *head end*, cada módem puede calcular el momento en que en realidad se inició la primera minirranura. La longitud de la minirranura depende de la red. Una carga útil típica es de 8 bytes.

Durante la inicialización, el amplificador *head end* también asigna a cada módem una minirranura a fin de utilizarla para solicitar el ancho de banda ascendente. Como regla, la misma minirranura se asignará a múltiples módems, lo que produce contención por las minirranuras. Cuando una computadora necesita enviar un paquete, lo transfiere al módem, el cual a continuación solicita el número necesario de minirranuras para realizar el envío. Si la solicitud es aceptada, el amplificador *head end* coloca una confirmación de recepción en el canal descendente que indica al módem cuáles minirranuras se han reservado para su paquete. A continuación el paquete se envía, comenzando en la minirranura asignada para ese propósito. Es posible solicitar paquetes adicionales mediante el uso de un campo en el encabezado.

Por otro lado, si hay contienda por la minirranura solicitada, no habrá confirmación de recepción y el módem simplemente espera un tiempo aleatorio y vuelve a intentar. Después de cada falla sucesiva, el tiempo aleatorio se duplica. (Para los lectores que ya están familiarizados con la conectividad de redes, este algoritmo sólo es ALOHA ranurado con retroceso exponencial binario. No es posible utilizar Ethernet en el cable porque las estaciones no pueden detectar el medio. En el capítulo 4 retomaremos este tema.)

Los canales descendentes se manejan de manera diferente que los ascendentes. Por un lado, sólo hay un emisor (el amplificador *head end*) por lo que no hay contienda ni necesidad de minirranuras, lo que en realidad es multiplexión estadística por división de tiempo. Por otro lado, el tráfico descendente por lo general es mayor que el ascendente, por lo que se utiliza un tamaño fijo de paquete de 204 bytes. Parte de esto es un código de corrección de errores Reed-Solomon y otra es sobrecarga, lo que deja una carga útil de usuario de 184 bytes. Estos números se eligieron por compatibilidad con la televisión digital que utiliza MPEG-2, por lo que los canales descendente de datos y de TV se formatean de la misma manera. Lógicamente, las conexiones son como se muestra en la figura 2-49.

Volviendo al tema de la inicialización, una vez que el módem ha terminado la alineación y obtenido su canal ascendente, canal descendente y sus asignaciones de minirranuras, puede comenzar

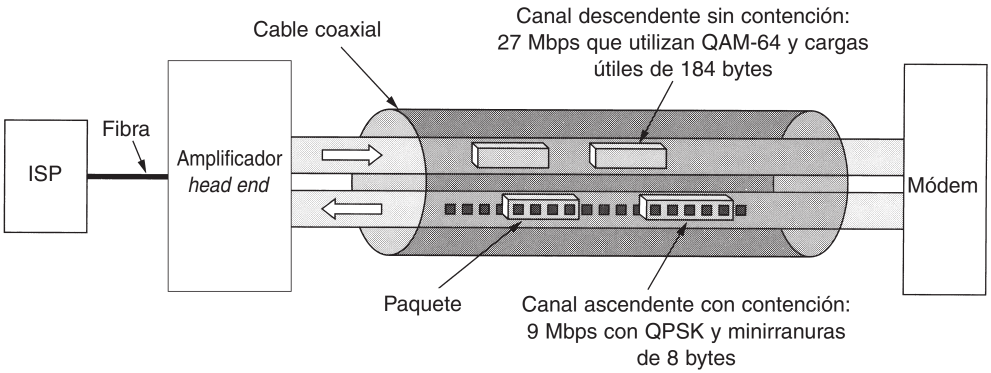


Figura 2-49. Detalles típicos de los canales ascendente y descendente en Norteamérica.

a enviar paquetes. El primer paquete que envía es uno para el ISP pidiéndole una dirección IP, que se asigna de manera dinámica utilizando un protocolo llamado DHCP, el cual analizaremos en el capítulo 5. También solicita y obtiene una hora precisa del amplificador *head end*.

El siguiente paso tiene que ver con la seguridad. Debido a que el cable es un medio compartido, cualquiera que desee utilizarlo puede leer todo el tráfico que pasa a través de él. Para evitar a los curiosos, todo el tráfico se encripta en ambas direcciones. Parte del proceso de inicialización involucra el establecimiento de claves de encriptación. Al principio se podría pensar que es imposible que dos extraños, el amplificador *head end* y el módem, establezcan una clave secreta a plena luz del día con miles de personas observando. Sin embargo, no lo es, pero tendremos que esperar hasta el capítulo 8 para explicar cómo puede hacerse (la respuesta corta sería: mediante el algoritmo de Diffie-Hellman).

Por último, el módem tiene que iniciar sesión y proporcionar su identificador único a través del canal seguro. Hasta este punto la inicialización está completa. A continuación, el usuario puede iniciar sesión en el ISP y comenzar a trabajar.

Hay mucho más por decir acerca de los módems de cable. Algunas referencias son (Adams y Dulchinos, 2001; Donaldson y Jones, 2001, y Dutta-Roy, 2001).

2.7.5 ADSL en comparación con el cable

¿Qué es mejor, ADSL o el cable? Eso es como preguntar cuál sistema operativo es mejor. O qué lenguaje es mejor. O qué religión. La respuesta que obtenga depende de a quién le pregunte. Comparemos ADSL y el cable mediante algunos puntos. Los dos utilizan la fibra óptica en la red dorsal, pero difieren en el extremo. El cable utiliza cable coaxial; ADSL, cable de par trenzado. La capacidad de carga teórica del cable coaxial es de cientos de veces más que el cable de par trenzado. Sin embargo, la capacidad máxima del cable no está disponible para los usuarios de datos porque la mayor parte del ancho de banda del cable se desperdicia en cosas inútiles; por ejemplo, en programas de televisión.

En la práctica, es difícil generalizar acerca de la capacidad efectiva. Los proveedores de ADSL indican específicamente el ancho de banda (por ejemplo, flujo descendente de 1 Mbps, flujo ascendente de 256 kbps) y por lo general logran alrededor de 80% de manera consistente. Los proveedores de cable no dan ninguna indicación pues la capacidad efectiva depende de cuántas personas estén actualmente activas en el segmento de cable del usuario. Algunas veces puede ser mejor que ADSL y otras podría ser peor. Sin embargo, lo que sí puede ser molesto es la incertidumbre. Tener servicio excelente por un minuto no garantiza que al siguiente minuto también lo tendrá, debido a que el ancho de banda más grande de la ciudad ha sido acaparado por otra computadora.

Conforme un sistema ADSL adquiere usuarios, este incremento tiene muy poco efecto en los usuarios existentes, debido a que cada usuario tiene una conexión dedicada. Con el cable, conforme más personas se suscriban al servicio de Internet, el rendimiento de los usuarios existentes disminuirá. El único remedio es que el operador de cable divida los cables ocupados y conecte en forma directa cada uno a un nodo de fibra óptica. Esto cuesta tiempo y dinero, y son presiones de negocios que se deben evitar.

Además, estudiamos otro sistema con un canal compartido como el cable: el sistema telefónico móvil. Aquí un grupo de usuarios también comparte una cantidad fija de ancho de banda. Por lo general, la FDM y la TDM lo dividen estrictamente en porciones fijas entre los usuarios activos pues el tráfico de voz es suave. Pero para el tráfico de datos, esta división rígida es muy ineficiente debido a que los usuarios de datos por lo general están inactivos, en cuyo caso el ancho de banda reservado es un desperdicio. No obstante, en este caso el acceso al cable es más parecido al sistema telefónico móvil que al sistema fijo.

La disponibilidad es un tema en el que ADSL y el cable difieren. Todas las personas tienen teléfono, pero no todos los usuarios están lo suficientemente cerca de su oficina central local para obtener ADSL. Por otro lado, no todos los usuarios tienen cable, pero si usted tiene cable y la compañía proporciona acceso a Internet, puede obtenerlo. Para el nodo de fibra o el amplificador *head end* la distancia no es un problema. También vale la pena mencionar que debido a que el cable inició como un medio de distribución de televisión, pocos negocios cuentan con él.

Debido a que es un medio de punto a punto, ADSL es inherentemente más seguro que el cable. Cualquier usuario de cable puede leer fácilmente todos los paquetes que pasen por el cable. Por esta razón, cualquier proveedor de cable que se precie de serlo encriptará todo el tráfico en ambas direcciones. Sin embargo, el hecho de que su vecino pueda obtener sus mensajes encriptados aún es menos seguro que el hecho de que no obtenga nada.

El sistema telefónico por lo general es más confiable que el cable. Por ejemplo, tiene energía reservada y continúa trabajando de manera normal incluso durante una falla en la energía. Con el cable, si falla la energía de cualquier amplificador de la cadena, todos los usuarios descendentes experimentarán un corte de manera instantánea.

Por último, la mayoría de los proveedores ADSL ofrece una opción de ISPs. Algunas veces la ley los obliga a hacerlo. Éste no siempre es el caso con los operadores de cable.

La conclusión es que ADSL y el cable son tan parecidos como diferentes. Ofrecen servicios comparables y, conforme la competencia entre ellos se avive más, probablemente precios comparables.

2.8 RESUMEN

La capa física es la base de todas las redes. La naturaleza impone dos límites fundamentales a todos los canales, y esto determina su ancho de banda. Estos límites son el de Nyquist, que tiene que ver con los canales sin ruido, y el de Shannon, para canales con ruido.

Los medios de transmisión pueden ser guiados y no guiados. Los principales medios guiados son el cable de par trenzado, el cable coaxial y la fibra óptica. Los medios no guiados incluyen la radio, las microondas, el infrarrojo y los láseres a través del aire. Un sistema de comunicación prometedor es la comunicación por satélite, especialmente los sistemas LEO.

Un elemento clave de la mayor parte de las redes de área amplia es el sistema telefónico. Sus componentes principales son los circuitos locales, troncales y conmutadores. Los circuitos locales son circuitos de cable de par trenzado, analógicos, que requieren módems para transmitir datos digitales. ADSL ofrece velocidades de hasta 50 Mbps dividiendo el circuito local en muchos canales individuales y modulando cada uno por separado. Los ciclos locales inalámbricos son otro nuevo desarrollo que observar, especialmente LMDS.

Las troncales son digitales y se pueden multiplexar de varias formas, incluidas FDM, TDM y WDM. Tanto la conmutación de circuitos como la de paquetes son importantes.

Para las aplicaciones móviles, el sistema de teléfono fijo no es adecuado. En la actualidad los teléfonos móviles se están usando ampliamente para voz y muy pronto se utilizarán ampliamente para datos. La primera generación fue analógica, y dominada por AMPS. La segunda generación fue digital, en la que D-AMPS, GSM y CDMA eran las opciones principales. La tercera generación será digital y se basará en la banda ancha CDMA.

Un sistema alternativo para acceso a red es el sistema de televisión por cable, que ha evolucionado de manera gradual de una antena comunal a una red híbrida de fibra óptica y cable coaxial. Potencialmente, ofrece un ancho de banda muy alto, pero en la práctica, el ancho de banda real disponible depende del número de usuarios activos y de lo que estén haciendo.

PROBLEMAS

1. Calcule los coeficientes de Fourier para la función $f(\mathrm{t}) = \mathrm{t}$ $(0 \le t \le 1)$.
2. Un canal sin ruido de 4 kHz se muestrea cada 1 mseg. ¿Cuál es la tasa de datos máxima?
3. Los canales de televisión tienen un ancho de 6 Mhz. ¿Cuántos bits/seg se pueden enviar si se usan señales digitales de cuatro niveles? Suponga que el canal es sin ruido.
4. Si se envía una señal binaria por un canal de 3 kHz cuya relación de señal a ruido es de 20 dB, ¿cuál es la tasa de datos máxima que se puede obtener?
5. ¿Qué relación de señal a ruido se necesita para poner una portadora T1 en una línea de 50 kHz?

6. ¿Qué diferencia hay entre una estrella pasiva y un repetidor activo en una red de fibra óptica?
7. ¿Cuánto ancho de banda existe en 0.1 micras de espectro a una longitud de onda de 1 micra?
8. Se desea enviar una secuencia de imágenes de pantalla de computadora por una fibra óptica. La pantalla es de 480×640 píxeles y cada píxel ocupa 24 bits. Hay 60 imágenes de pantalla por segundo. ¿Cuánto ancho de banda se necesita y cuántas micras de longitud de onda se necesitan para esta banda a 1.30 micras?
9. ¿Se cumple el teorema de Nyquist para la fibra óptica o solamente para el alambre de cobre?
10. En la figura 2-6 la banda de la izquierda es más angosta que las otras. ¿Por qué?
11. A menudo las antenas de radio funcionan mejor cuando el diámetro de la antena es igual a la longitud de la onda de radio. Las antenas prácticas tienen diámetros desde 1 cm hasta 5 m de diámetro. ¿Qué rango de frecuencias cubre esto?
12. El desvanecimiento por múltiples trayectorias alcanza un máximo cuando los dos haces llegan desfasados 180 grados. ¿Qué tan diferentes deben ser las trayectorias para que el desvanecimiento sea máximo para un enlace de microondas de 1 GHz de 50 km de largo?
13. Un rayo láser de 1 mm de diámetro se apunta a un detector de 1 mm de diámetro a 100 m en el techo de un edificio. ¿Cuánta desviación angular deberá tener el láser antes de que pierda al detector?
14. Los 66 satélites de órbita baja en el proyecto Iridium se dividen en seis collares alrededor de la Tierra. A la altitud que están utilizando, el periodo es de 90 minutos. ¿Cuál es el intervalo promedio de transferencias de celdas para un transmisor fijo?
15. Considere un satélite a una altitud de satélites geoestacionarios pero cuyo plan de órbitas se inclina hacia el plano ecuatorial a un ángulo ϕ. Para un usuario fijo en la superficie de la Tierra a una altitud norte ϕ, ¿este satélite da la impresión en el cielo de que no tiene movimiento? De lo contrario, describa su movimiento.
16. Cuántos códigos de oficina central local había antes de 1984, cuando cada oficina central tenía el nombre de los tres dígitos de su código de área y los primeros tres dígitos del número local? Los códigos de área iniciaban con un dígito en el rango de 2–9, tenían un 0 o un 1 como su segundo dígito, y terminaban con cualquier dígito. Los primeros dos dígitos de un número local siempre estaban en el rango de 2–9. El tercer dígito podía ser cualquiera.
17. Utilizando *sólo* los datos dados en el texto, ¿cuál es la cantidad máxima de teléfonos que el sistema existente de Estados puede manejar sin cambiar el plan de numeración o agregar equipo adicional? ¿Es posible alcanzar esta cantidad de teléfonos? Para propósitos de este problema, una computadora o máquina de fax cuenta como un teléfono. Suponga que sólo hay un dispositivo por línea de suscriptor.
18. Un sistema telefónico simple consiste en dos oficinas centrales locales y una interurbana a la que está conectada cada oficina central por una troncal dúplex de 1 MHz. En promedio, cada teléfono se usa para hacer cuatro llamadas por cada jornada de 8 horas. La duración media de las llamadas es de 6 minutos. El 10% de las llamadas son de larga distancia (esto es, pasan por la oficina interurbana). ¿Cuál es la cantidad máxima de teléfonos que puede manejar una oficina central local? (Suponga que hay 4 kHz por circuito.)
19. Una compañía de teléfonos regional tiene 10 millones de suscriptores. Cada uno de sus teléfonos está conectado a una oficina central local mediante un cable de par trenzado de cobre. La longitud promedio

de estos cables de par trenzado es de 10 km. ¿Cuánto vale el cobre de los circuitos locales? Suponga que la sección transversal de cada filamento es un círculo de 1 mm de diámetro, que el peso específico relativo del cobre es 9.0 y que el cobre se vende a 3 dólares por kilogramo.

20. ¿Un gasoducto es un sistema símplex, uno semidúplex, uno dúplex total, o ninguno de los antes mencionados?

21. El costo de un microprocesador potente se ha reducido a tal grado que ahora es posible incluir uno en cada módem. ¿Cómo afecta esto el manejo de errores en las líneas telefónicas?

22. Un diagrama de constelación de módem, similar al de la figura 2-25, tiene puntos de datos en las siguientes coordenadas: (1, 1), (1, −1), (−1, 1) y (−1, −1). ¿Cuántos bps puede lograr un módem a 1200 baudios con estos parámetros?

23. Un diagrama de constelación de módem, similar al de la figura 2-25, tiene puntos de datos en (0, 1) y (0, 2). ¿El módem usa modulación de fase o modulación de amplitud?

24. En un diagrama de constelación todos los puntos están en un círculo centrado en el origen. ¿Qué tipo de modulación se utiliza?

25. ¿Cuántas frecuencias utiliza un módem QAM-64 de dúplex total?

26. Un sistema ADSL que utiliza DMT asigna 3/4 de los canales de datos disponibles al enlace descendente. Utiliza modulación QAM-64 en cada canal. ¿Cuál es la capacidad del enlace descendente?

27. En el ejemplo de cuatro sectores LMDS de la figura 2-30, cada sector tiene su propio canal de 36 Mbps. De acuerdo con la teoría de encolamiento, si el canal está cargado en 50%, el tiempo de encolamiento será igual que el de descarga. Bajo estas condiciones, ¿cuánto tiempo se tarda en bajar una página Web de 5 KB? ¿Cuánto tiempo se tarda en bajar la página a través de una línea ADSL de 1 Mbps? ¿A través de un módem de 56 kbps?

28. Diez señales, cada una de las cuales requiere 4000 Hz, se multiplexan en un solo canal utilizando FDM. ¿Cuál es el ancho de banda mínimo requerido para el canal multiplexado? Suponga que las bandas de protección tienen un ancho de 400 Hz.

29. ¿Por qué se fijó el tiempo de muestreo de PCM en 125 µseg?

30. ¿Cuál es el porcentaje de sobrecarga en una portadora T1?; esto es, ¿qué porcentaje de los 1.544 Mbps no se entrega al usuario final?

31. Compare la tasa de datos máxima de un canal sin ruido de 4 kHz que utiliza:
(a) Codificación analógica con 2 bits por muestra.
(b) El sistema T1 de PCM.

32. Si un sistema de portador T1 pierde la pista de dónde está, trata de resincronizarse con base en el primer bit de cada trama. ¿Cuántas tramas se tendrían que inspeccionar en promedio para resincronizarse con una probabilidad de 0.001 de estar en un error?

33. ¿Cuál es la diferencia, si la hay, entre la parte desmoduladora de un módem y la parte codificadora de un codec? (Después de todo, ambas convierten señales analógicas a digitales.)

34. Una señal se transmite en forma digital por un canal sin ruido de 4 kHz, con una muestra cada 125 µseg. ¿Cuántos bits por segundo se envían realmente con cada uno de los siguientes métodos de codificación?
(a) CCITT, 2.048 Mbps estándar.

(b) DPCM con un valor de señal relativo de 4 bits.
(c) Modulación delta.

35. Se codifica una onda senoidal pura de amplitud *A* usando modulación delta, con *x* muestras/seg. Una salida de +1 corresponde a un cambio de señal de $+A/8$, y una señal de salida de –1 corresponde a un cambio de señal de $-A/8$. ¿Cuál es la frecuencia más alta que se puede rastrear sin error acumulativo?

36. Los relojes de SONET tienen una tasa de arrastre de casi 1 parte en 10^9. ¿Cuánto tiempo tomará para que el arrastre iguale el ancho de 1 bit? ¿Cuáles son las implicaciones de este cálculo?

37. En la figura 2-37 se establece que la tasa de datos del usuario para OC-3 es de 148.608 Mbps. Demuestre cómo se puede deducir esta cantidad de los parámetros de OC-3 de SONET.

38. Para acomodar tasas de datos menores que STS-1, SONET tiene un sistema de tributarias virtuales (VT). Una VT es una carga útil parcial que se puede insertar en una trama STS-1 y combinar con otras cargas útiles parciales para llenar la trama de datos. VT1.5 utiliza 3 columnas, VT2 utiliza 4, VT3 utiliza 6 y VT6 utiliza 12 columnas de una trama STS-1. ¿Cuál VT puede acomodar
(a) Un servicio DS-1 (1.544 Mbps)?
(b) Un servicio europeo CEPT-1 (2.048 Mbps o E1)?
(c) Un servicio DS-2 (6.312 Mbps)?

39. ¿Cuál es la diferencia esencial entre la conmutación de mensajes y la de paquetes?

40. ¿Cuál es el ancho de banda disponible para el usuario en una conexión OC-12c?

41. Tres redes de conmutación de paquetes contienen *n* nodos cada una. La primera red tiene una topología de estrella con un conmutador central, la segunda es un anillo (bidireccional) y la tercera está interconectada por completo, con una conexión de cada nodo hacia cada uno de los otros nodos. ¿Cuáles son las rutas de transmisión óptima, media y de peor caso en saltos?

42. Compare el retardo al enviar un mensaje de *x* bits por una trayectoria de *k* saltos en una red de conmutación de circuitos y en una red de conmutación de paquetes (con carga ligera). El tiempo de establecimiento de circuito es de *s* segundos, el retardo de propagación es de *d* segundos por salto, el tamaño del paquete es de *p* bits y la tasa de datos es de *b* bps. ¿En qué condiciones tiene un retardo menor la red de paquetes?

43. Suponga que se van a transmitir *x* bits de datos de usuario por una trayectoria de *k* saltos en una red de conmutación de paquetes como una serie de paquetes, cada uno contiene *p* bits de datos y *h* bits de encabezado, donde $x \gg p + h$. La tasa de bits de las líneas es de *b* bps y el retardo de propagación es nulo. ¿Qué valor de *p* minimiza el retardo total?

44. En un sistema de telefónico móvil típico con celdas hexagonales se permite reutilizar una banda de frecuencia en una celda adyacente. Si están disponibles 840 frecuencias, ¿cuántas se pueden utilizar en una celda dada?

45. El diseño real de las celdas rara vez es tan regular como se muestra en la figura 2-41. Incluso la forma de las celdas individuales por lo general es irregular. Dé una posible razón de por qué sucedería esto.

46. Realice una estimación aproximada de la cantidad de microceldas PCS con un diámetro de 100 m que se requerirían para cubrir San Francisco (120 km^2).

47. Algunas veces cuando un usuario móvil cruza el límite de una celda a otra, la llamada actual se termina de manera repentina, aunque todos los transmisores y receptores estén funcionando correctamente. ¿Por qué?

48. D-AMPS tiene evidentemente una calidad de voz menor que GSM. ¿Ésta es la razón por la que D-AMPS necesita tener compatibilidad hacia atrás con AMPS, y GSM no? Si no es así, explique la causa.

49. Calcule el número máximo de usuarios que D-AMPS puede manejar de manera simultánea dentro de una celda. Realice el mismo cálculo para GSM. Explique la diferencia.

50. Suponga que *A*, *B* y *C*, transmiten de manera simultánea bits 0 mediante un sistema CDMA con las secuencias de chips que se muestran en la figura 2-45(b). ¿Cuál es la secuencia de chips resultante?

51. En el análisis acerca de la ortogonalidad de las secuencias de chips CDMA se dijo que si $\mathbf{S}\bullet\mathbf{T} = 0$, entonces $\mathbf{S}\bullet\overline{\mathbf{T}}$ también es 0. Pruebe esto.

52. Considere una manera diferente de mirar la propiedad de ortogonalidad de las secuencias de chips CDMA. Cada bit en un par de secuencias puede o no coincidir. Exprese la propiedad de ortogonalidad en términos de coincidencias y falta de coincidencias.

53. Un receptor CDMA obtiene los siguientes chips: (−1 +1 −3 +1 −1 −3 +1 +1). Suponiendo las secuencias de chips definidas en la figura 2-45(b), ¿cuáles estaciones transmitieron y qué bits envió cada una?

54. En su parte más baja, el sistema telefónico tiene forma de estrella, y todos los circuitos locales de un vecindario convergen en una oficina central local. En contraste, la televisión por cable consiste en un solo cable largo que pasa por todas las casas del mismo vecindario. Suponga que un cable de TV fuera de fibra óptica de 10 Gbps en lugar de cable de cobre. ¿Podría utilizarse para simular un modelo telefónico en el que todo mundo tuviera su propia línea privada a la oficina central local? Si esto fuera posible, ¿cuántas casas con un teléfono podrían conectarse a una sola fibra óptica?

55. Un sistema de TV por cable tiene cien canales comerciales y todos ellos alternan programas con anuncios. ¿Esto es más parecido a TDM o a FDM?

56. Una compañía de cable decide proporcionar acceso a Internet a través de cable en un vecindario que consiste en 5000 casas. La compañía utiliza cable coaxial y asignación de espectro que permite un ancho de banda descendente de 100 Mbps por cable. Para atraer clientes la compañía decide garantizar un ancho de banda descendente de por lo menos 2 Mbps a cada casa en cualquier momento. Describa lo que necesita hacer la compañía de cable para proporcionar esta garantía.

57. Tomando en cuenta la asignación espectral mostrada en la figura 2-48 y la información dada en el texto, ¿cuántos Mbps necesita asignar el sistema por cable al flujo ascendente y cuántos al flujo descendente?

58. ¿Qué tan rápido un usuario de cable puede recibir datos si la red está inactiva?

59. Multiplexar flujos de datos múltiples STS-1, llamados tributarias, tiene un papel importante en SONET. Un multiplexor 3:1 multiplexa tres tributarias STS-1 de entrada en un flujo STS-3 de salida. Esta multiplexión se realiza byte por byte, es decir, los tres primeros bytes de salida son los primeros bytes de las tributarias 1, 2 y 3, respectivamente. Los siguientes tres bytes de salida son los segundos bytes

de las tributarias 1, 2 y 3, respectivamente, etcétera. Escriba un programa que simule este multiplexor 3:1. El programa deberá consistir de cinco procesos. El proceso principal crea cuatro procesos, uno para cada una de las tres tributarias STS-1 y uno para el multiplexor. Cada proceso tributario lee una trama STS-1 de un archivo de entrada como una secuencia de 810 bytes. Tales procesos tributarios envían sus tramas (byte por byte) al proceso multiplexor. Éste recibe los bytes y envía una trama STS-3 (byte por byte) escribiéndola en una salida estándar. Utilice canalizaciones para la comunicación entre procesos.

3

LA CAPA DE ENLACE DE DATOS

En este capítulo estudiaremos los principios de diseño de la capa 2, la capa de enlace de datos. Este estudio tiene que ver con los algoritmos para lograr una comunicación confiable y eficiente entre dos máquinas adyacentes en la capa de enlace de datos. Por adyacente, queremos decir que las dos máquinas están conectadas por un canal de comunicaciones que actúa de manera conceptual como un alambre (por ejemplo, un cable coaxial, una línea telefónica o un canal inalámbrico de punto a punto). La propiedad esencial de un canal que lo hace asemejarse a un alambre es que los bits se entregan con exactitud en el mismo orden en que fueron enviados.

A primera vista podría pensarse que este problema es tan trivial que no hay ningún software que estudiar: la máquina *A* sólo pone los bits en el alambre, y la máquina *B* simplemente los toma. Por desgracia, los circuitos de comunicación cometen errores ocasionales. Además, tienen una tasa de datos finita y hay un retardo de propagación diferente de cero entre el momento en que se envía un bit y el momento en que se recibe. Estas limitaciones tienen implicaciones importantes para la eficiencia de la transferencia de datos. Los protocolos usados para comunicaciones deben considerar todos estos factores. Dichos protocolos son el tema de este capítulo.

Tras una introducción a los aspectos clave de diseño presentes en la capa de enlace de datos, comenzaremos nuestro estudio de sus protocolos observando la naturaleza de los errores, sus causas y la manera en que se pueden detectar y corregir. Después estudiaremos una serie de protocolos de complejidad creciente, cada uno de los cuales resuelve los problemas presentes en esta capa. Por último, concluiremos con un estudio del modelado y la corrección de los protocolos y daremos algunos ejemplos de protocolos de enlace de datos.

3.1 CUESTIONES DE DISEÑO DE LA CAPA DE ENLACE DE DATOS

La capa de enlace de datos tiene que desempeñar varias funciones específicas, entre las que se incluyen:

1. Proporcionar una interfaz de servicio bien definida con la capa de red.
2. Manejar los errores de transmisión.
3. Regular el flujo de datos para que receptores lentos no sean saturados por emisores rápidos.

Para cumplir con estas metas, la capa de enlace de datos toma de la capa de red los paquetes y los encapsula en **tramas** para transmitirlos. Cada trama contiene un encabezado, un campo de carga útil (*payload*) para almacenar el paquete y un terminador o final, como se ilustra en la figura 3-1. El manejo de las tramas es la tarea primordial de la capa de enlace de datos. En las siguientes secciones examinaremos en detalle todos los aspectos mencionados.

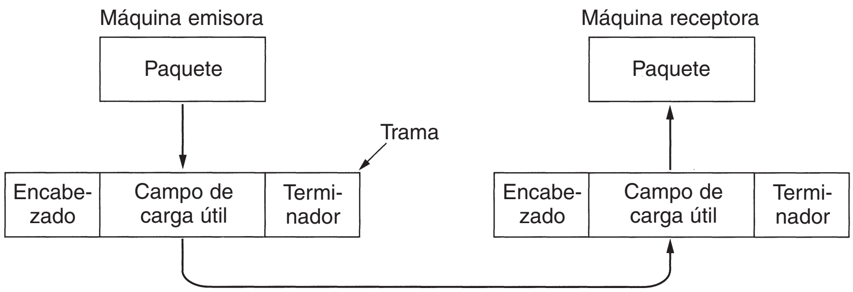


Figura 3-1. Relación entre los paquetes y las tramas.

Aunque este capítulo sólo analiza la capa de enlace de datos y los protocolos de enlace de datos, muchos de los principios que analizaremos aquí, como el control de errores y el de flujo, también se encuentran en la capa de transporte y en otros protocolos. De hecho, en muchas redes, estas funciones se encuentran sólo en las capas superiores y no en la de enlace de datos. Sin embargo, independientemente de donde se encuentren, los principios son casi los mismos, por lo que en realidad no importa en qué parte del libro los analicemos. Por lo general, éstos se muestran en la capa de enlace de datos en sus formas más simples y puras, por lo que dicha capa es un buen lugar para examinarlos en detalle.

3.1.1 Servicios proporcionados a la capa de red

La función de la capa de enlace de datos es suministrar servicios a la capa de red. El servicio principal es transferir datos de la capa de red en la máquina de origen a la capa de red en la máquina de destino. En la capa de red de la máquina de origen hay una entidad, llamada proceso, que entrega algunos bits a la capa de enlace de datos para transmitirlos a la máquina de destino. El tra-

bajo de la capa de enlace de datos es transmitir los bits a la máquina de destino, para que puedan ser entregados a su capa de red, como se muestra en la figura 3-2(a). La transmisión real sigue la trayectoria de la figura 3-2(b), pero es más fácil pensar en términos de dos procesos de capa de enlace de datos que se comunican usando un protocolo de enlace de datos. Por esta razón, a lo largo de este capítulo usaremos de manera implícita el modelo de la figura 3-2(a).

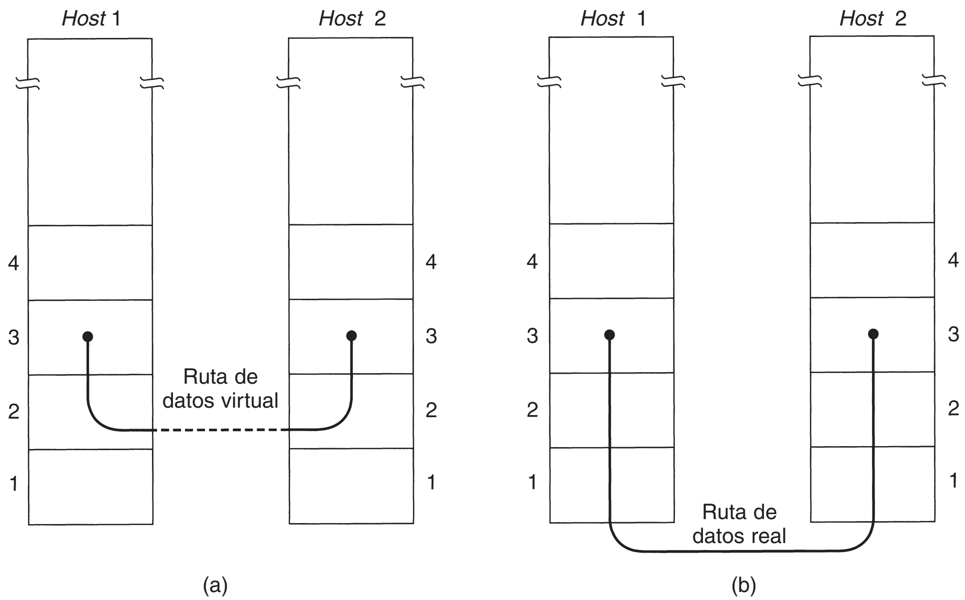


Figura 3-2. (a) Comunicación virtual. (b) Comunicación real.

La capa de enlace de datos puede diseñarse para ofrecer varios servicios. Los servicios reales ofrecidos pueden variar de sistema a sistema. Tres posibilidades razonables que normalmente se proporcionan son:

1. Servicio no orientado a la conexión sin confirmación de recepción.
2. Servicio no orientado a la conexión con confirmación de recepción.
3. Servicio orientado a la conexión con confirmación de recepción.

Consideremos cada uno de ellos por separado.

El servicio no orientado a la conexión sin confirmación de recepción consiste en hacer que la máquina de origen envíe tramas independientes a la máquina de destino sin pedir que ésta confirme la recepción. No se establece conexión de antemano ni se libera después. Si se pierde una trama debido a ruido en la línea, en la capa de enlace de datos no se realiza ningún intento por detectar la pérdida ni por recuperarse de ella. Esta clase de servicio es apropiada cuando la tasa de errores es muy baja, por lo que la recuperación se deja a las capas superiores. También es apropiada para el tráfico en tiempo real, por ejemplo de voz, en el que la llegada retrasada de datos es peor que

los errores de datos. La mayoría de las LANs utilizan servicios no orientados a la conexión sin confirmación de recepción en la capa de enlace de datos.

El siguiente paso hacia adelante en cuanto a confiabilidad es el servicio no orientado a la conexión con confirmación de recepción. Cuando se ofrece este servicio tampoco se utilizan conexiones lógicas, pero se confirma de manera individual la recepción de cada trama enviada. De esta manera, el emisor sabe si la trama ha llegado bien o no. Si no ha llegado en un tiempo especificado, puede enviarse nuevamente. Este servicio es útil en canales inestables, como los de los sistemas inalámbricos.

Tal vez valga la pena poner énfasis en que proporcionar confirmaciones de recepción en la capa de enlace de datos sólo es una optimización, nunca un requisito. La capa de red siempre puede enviar un paquete y esperar que se confirme su recepción. Si la confirmación no llega antes de que expire el temporizador, el emisor puede volver a enviar el mensaje. El problema con esta estrategia es que las tramas tienen una longitud máxima impuesta por el hardware mientras que los paquetes de la capa de red no la tienen. Si el paquete promedio se divide en, digamos, 10 tramas, y se pierde 20% de todas las tramas enviadas, el paquete puede tardar mucho tiempo en pasar. Si las tramas se confirman y retransmiten de manera individual, los paquetes completos pasan con mayor rapidez. En los canales confiables, como la fibra óptica, la sobrecarga que implica el uso de un protocolo de enlace de datos muy robusto puede ser innecesaria, pero en canales inalámbricos bien vale la pena el costo debido a su inestabilidad inherente.

Regresando a nuestros servicios, el servicio más refinado que puede proporcionar la capa de enlace de datos a la capa de red es el servicio orientado a la conexión. Con este servicio, las máquinas de origen y de destino establecen una conexión antes de transferir datos. Cada trama enviada a través de la conexión está numerada, y la capa de enlace de datos garantiza que cada trama enviada llegará a su destino. Es más, garantiza que cada trama será recibida exactamente una vez y que todas las tramas se recibirán en el orden adecuado. En contraste, con el servicio no orientado a la conexión es posible que una confirmación de recepción perdida cause que una trama se envíe varias veces y, por lo tanto, que se reciba varias veces. Por su parte, el servicio orientado a la conexión proporciona a los procesos de la capa de red el equivalente de un flujo de bits confiable.

Cuando se utiliza un servicio orientado a la conexión, las transferencias tienen tres fases distintas. En la primera, la conexión se establece haciendo que ambos lados inicialicen las variables y los contadores necesarios para seguir la pista de las tramas que han sido recibidas y las que no. En la segunda fase se transmiten una o más tramas. En la tercera fase, la conexión se cierra y libera las variables, los búferes y otros recursos utilizados para mantener la conexión.

Considere un ejemplo típico: una subred de WAN que consiste en enrutadores conectados por medio de líneas telefónicas alquiladas de punto a punto. Cuando llega una trama a un enrutador, el hardware la examina para verificar si está libre de errores (mediante una técnica que veremos más adelante en este capítulo), y después la pasa al software de la capa de enlace de datos (que podría estar integrado en un *chip* de la tarjeta de interfaz de red). Dicho software comprueba si ésta es la trama esperada y, de ser así, entrega el paquete contenido en el campo de carga útil al software de enrutamiento. A continuación, este software elige la línea de salida adecuada y reenvía el paquete al software de la capa de enlace de datos, que luego lo transmite. En la figura 3-3 se muestra el flujo a través de dos enrutadores.

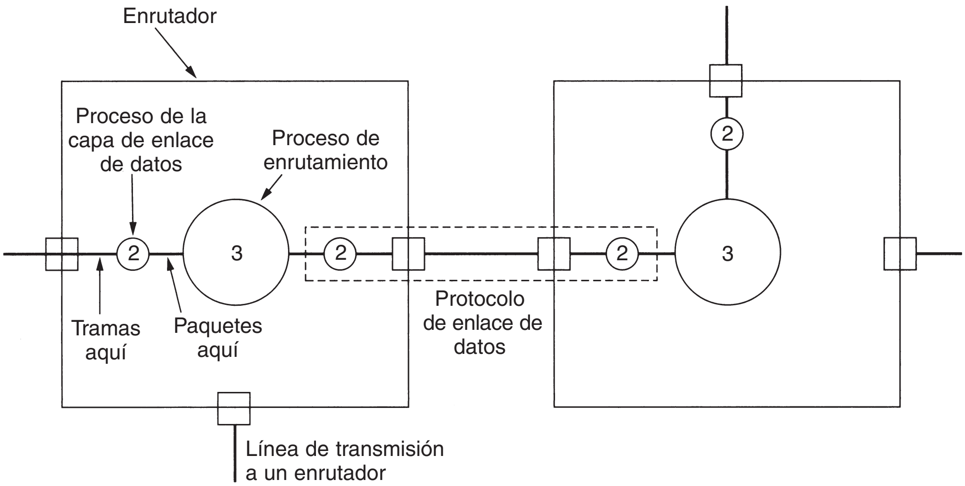


Figura 3-3. Ubicación del protocolo de enlace de datos.

El código de enrutamiento con frecuencia requiere que el trabajo se haga bien, es decir, que haya conexiones estables y ordenadas en cada una de las líneas punto a punto. No quiere que se le moleste frecuentemente con paquetes que se perdieron en el camino. Es responsabilidad del protocolo de enlace de datos, mostrado en el rectángulo punteado, hacer que las líneas de comunicación no estables parezcan perfectas o, cuando menos, bastante buenas. Como información adicional, aunque hemos mostrado múltiples copias del software de la capa de enlace de datos en cada enrutador, de hecho una sola copia maneja todas las líneas, con diferentes tablas y estructuras de datos para cada una.

3.1.2 Entramado

A fin de proporcionar servicios a la capa de red, la de enlace de datos debe utilizar los servicios que la capa física le proporciona. Lo que hace la capa física es aceptar un flujo de bits puros e intentar entregarlo al destino. No se garantiza que este flujo de bits esté libre de errores. La cantidad de bits recibidos puede ser menor, igual o mayor que la cantidad de bits transmitidos, y éstos pueden tener diferentes valores. Es responsabilidad de la capa de enlace de datos detectar y, de ser necesario, corregir los errores.

El método común es que la capa de enlace de datos divida el flujo de bits en tramas separadas y que calcule la suma de verificación de cada trama. (Posteriormente en este capítulo se analizarán los algoritmos de suma de verificación.) Cuando una trama llega al destino, se recalcula la suma de verificación. Si la nueva suma de verificación calculada es distinta de la contenida en la trama, la capa de enlace de datos sabe que ha ocurrido un error y toma medidas para manejarlo (por ejemplo, descartando la trama mala y, posiblemente, regresando un informe de error).

La división en tramas del flujo de bits es más difícil de lo que parece a primera vista. Una manera de lograr esta división en tramas es introducir intervalos de tiempo entre las tramas, de la misma manera que los espacios entre las palabras en el texto común. Sin embargo, las redes pocas veces ofrecen garantías sobre la temporización, por lo que es posible que estos intervalos sean eliminados o que puedan introducirse otros intervalos durante la transmisión.

Puesto que es demasiado riesgoso depender de la temporización para marcar el inicio y el final de cada trama, se han diseñado otros métodos. En esta sección veremos cuatro métodos:

1. Conteo de caracteres.
2. Banderas, con relleno de caracteres.
3. Banderas de inicio y fin, con relleno de bits.
4. Violaciones de codificación de la capa física.

El primer método de entramado se vale de un campo en el encabezado para especificar el número de caracteres en la trama. Cuando la capa de enlace de datos del destino ve la cuenta de caracteres, sabe cuántos caracteres siguen y, por lo tanto, dónde está el fin de la trama. Esta técnica se muestra en la figura 3-4(a) para cuatro tramas de 5, 5, 8 y 8 caracteres de longitud, respectivamente.

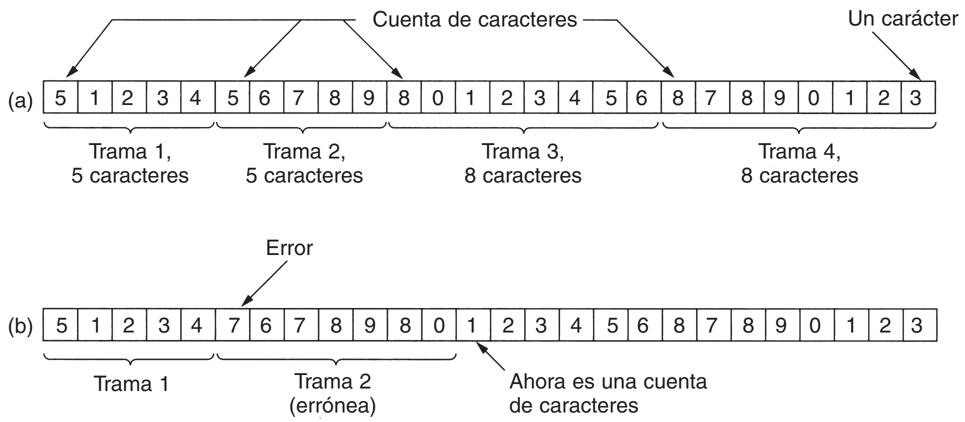


Figura 3-4. Un flujo de caracteres. (a) Sin errores. (b) Con un error.

El problema con este algoritmo es que la cuenta puede alterarse por un error de transmisión. Por ejemplo, si la cuenta de caracteres de 5 en la segunda trama de la figura 3-4(b) se vuelve un 7, el destino perderá la sincronía y será incapaz de localizar el inicio de la siguiente trama. Incluso si el destino sabe que la trama está mal porque la suma de verificación es incorrecta, no tiene forma de saber dónde comienza la siguiente trama. Regresar una trama a la fuente solicitando una retransmisión tampoco ayuda, ya que el destino no sabe cuántos caracteres tiene que saltar para

llegar al inicio de la retransmisión. Por esta razón, en la actualidad casi no se utiliza el método de conteo de caracteres.

El segundo método de entramado evita el problema de tener que sincronizar nuevamente después de un error, haciendo que cada trama inicie y termine con bytes especiales. En el pasado, los bytes de inicio y final eran diferentes, pero en los años recientes la mayoría de los protocolos han utilizado el mismo byte, llamado **bandera** (o indicador), como delimitador de inicio y final, que en la figura 3-5(a) se muestra como FLAG. De esta manera, si el receptor pierde la sincronía, simplemente puede buscar la bandera para encontrar el final e inicio de la trama actual. Dos banderas consecutivas señalan el final de una trama y el inicio de la siguiente.

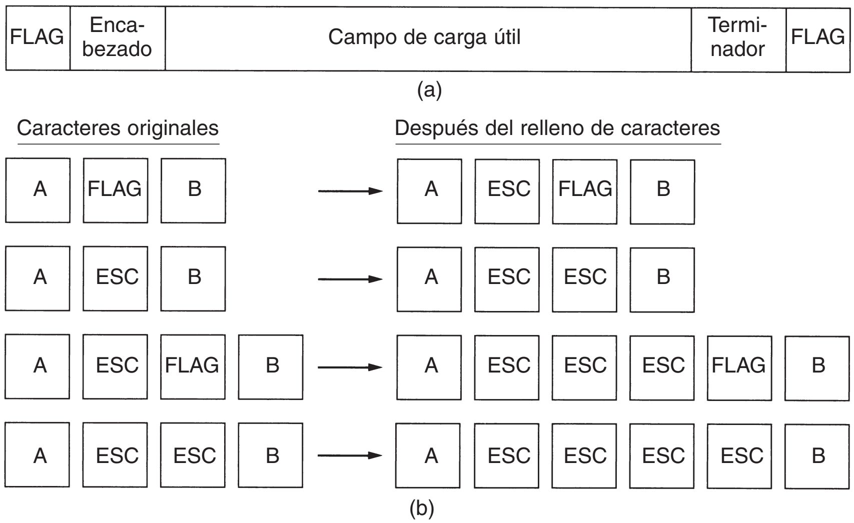


Figura 3-5. (a) Una trama delimitada por banderas. (b) Cuatro ejemplos de secuencias de bytes antes y después del relleno de caracteres.

Cuando se utiliza este método para transmitir datos binarios, como programas objeto o números de punto flotante, surge un problema serio. Se puede dar el caso con mucha facilidad de que el patrón de bits de la bandera aparezca en los datos (*payload*), lo que interferiría en el entramado. Una forma de resolver este problema es hacer que la capa de enlace de datos del emisor inserte un byte de escape especial (ESC) justo antes de cada bandera "accidental" en los datos. La capa de enlace de datos del lado receptor quita el byte de escape antes de entregar los datos a la capa de red. Esta técnica se llama **relleno de caracteres.** Por lo tanto, una bandera de entramado se puede distinguir de uno en los datos por la ausencia o presencia de un byte de escape que la antecede.

Por supuesto que surge la pregunta de qué sucede si un byte de escape aparece en medio de los datos. La respuesta es que también se rellena con un byte de escape. Por lo tanto, cualquier byte de escape individual es parte de una secuencia de escape, mientras que uno doble indica que un

escape sencillo apareció de manera natural en los datos. En la figura 3-5(b) se muestran algunos ejemplos. En todos los casos, la secuencia de bytes que se entrega después de la eliminación de los bytes de escape es exactamente la misma que la secuencia de bytes original.

El esquema de relleno de caracteres que se muestra en la figura 3-5 es una ligera simplificación del esquema empleado en el protocolo PPP que la mayoría de las computadoras utiliza para comunicarse con el proveedor de servicios de Internet. Más tarde analizaremos este protocolo.

Una desventaja importante del uso de esta técnica de entramado es que está fuertemente atada a los caracteres de 8 bits. No todos los códigos utilizan caracteres de 8 bits. Por ejemplo, UNICODE utiliza caracteres de 16 bits. A medida que se desarrollaron las redes, las desventajas de incorporar la longitud del código de caracteres en el mecanismo de entramado se volvieron más obvias, por lo que tuvo que desarrollarse una técnica nueva que permitiera caracteres de tamaño arbitrario.

La nueva técnica permite que las tramas de datos contengan un número arbitrario de bits y admite códigos de caracteres con un número arbitrario de bits por carácter. Dicha técnica funciona de la siguiente manera: cada trama comienza y termina con un patrón especial de bits, 01111110 (que es de hecho una bandera). Cada vez que la capa de enlace de datos del emisor encuentra cinco unos consecutivos en los datos, automáticamente inserta un bit 0 en el flujo de bits saliente. Este **relleno de bits** es análogo al relleno de caracteres, en el cual un byte de escape se inserta en el flujo de caracteres saliente antes de un byte igual a la bandera de entramado en los datos.

Cuando el receptor ve cinco bits 1 de entrada consecutivos, seguidos de un bit 0, automáticamente extrae (es decir, borra) el bit 0 de relleno. Así como el relleno de caracteres es completamente transparente para la capa de red en ambas computadoras, también lo es el relleno de bits. Si los datos de usuario contienen el patrón indicador 01111110, éste se transmite como 011111010, pero se almacena en la memoria del receptor como 01111110. En la figura 3-6 se da un ejemplo del relleno de bits.

(a) 011011111111111111110010

(b) 011011111011111011111010010

Bits de relleno

(c) 011011111111111111110010

Figura 3-6. Relleno de bits. (a) Los datos originales. (b) Los datos, según aparecen en la línea. (c) Los datos, como se guardan en la memoria del receptor tras eliminar el relleno.

Con el relleno de bits, el límite entre las dos tramas puede ser reconocido sin ambigüedades mediante el patrón de banderas. De esta manera, si el receptor pierde la pista de dónde está, todo lo que tiene que hacer es explorar la entrada en busca de secuencias de banderas, pues sólo pueden ocurrir en los límites de las tramas y nunca en los datos.

El último método de entramado sólo se aplica a las redes en las que la codificación en el medio físico contiene cierta redundancia. Por ejemplo, algunas LANs codifican un bit de datos usando dos bits físicos. Normalmente, un bit 1 es un par alto-bajo, y un bit 0 es un par bajo-alto. El esquema implica que cada bit de datos tiene una transición a medio camino, lo que hace fácil

para el receptor localizar los límites de los bits. Las combinaciones alto-alto y bajo-bajo no se usan para datos, pero en algunos protocolos se utilizan para delimitar tramas.

Como nota final sobre el entramado, muchos protocolos de enlace de datos usan, por seguridad, una combinación de cuenta de caracteres con uno de los otros métodos. Cuando llega una trama, se usa el campo de cuenta para localizar el final de la trama. Sólo si el delimitador apropiado está presente en esa posición y la suma de verificación es correcta, la trama se acepta como válida. De otra manera, se explora el flujo de entrada en busca del siguiente delimitador.

3.1.3 Control de errores

Una vez resuelto el problema de marcar el inicio y el final de cada trama, llegamos al siguiente problema: cómo asegurar que todas las tramas realmente se entreguen en el orden apropiado a la capa de red del destino. Suponga que el emisor se dedicó a enviar tramas sin importarle si estaban llegando de manera adecuada. Esto podría estar bien para un servicio no orientado a la conexión sin confirmación de recepción, pero no será correcto para un servicio confiable orientado a la conexión.

La manera normal de asegurar la entrega confiable de datos es proporcionar retroalimentación al emisor sobre lo que está ocurriendo en el otro lado de la línea. Por lo general, el protocolo exige que el receptor regrese tramas de control especiales que contengan confirmaciones de recepción positivas o negativas de las tramas que llegan. Si el emisor recibe una confirmación de recepción positiva de una trama, sabe que la trama llegó correctamente. Por otra parte, una confirmación de recepción negativa significa que algo falló y que la trama debe transmitirse otra vez.

Una complicación adicional surge de la posibilidad de que los problemas de hardware causen la desaparición de una trama completa (por ejemplo, por una ráfaga de ruido). En este caso, el receptor no reaccionará en absoluto, ya que no tiene razón para reaccionar. Debe quedar claro que un protocolo en el cual el emisor envía una trama y luego espera una confirmación de recepción, positiva o negativa, se quedaría esperando eternamente si se pierde por completo una trama debido a una falla de hardware.

Esta posibilidad se maneja introduciendo temporizadores en la capa de enlace de datos. Cuando el emisor envía una trama, por lo general también inicia un temporizador. Éste se ajusta de modo que expire cuando haya transcurrido un intervalo suficiente para que la trama llegue a su destino, se procese ahí y la confirmación de recepción se regrese al emisor. Por lo general, la trama se recibirá de manera correcta y la confirmación de recepción llegará antes de que el temporizador expire, en cuyo caso se cancelará.

Sin embargo, si la trama o la confirmación de recepción se pierden, el temporizador expirará, alertando al emisor sobre un problema potencial. La solución obvia es simplemente transmitir de nuevo la trama. Sin embargo, aunque las tramas pueden transmitirse muchas veces, existe el peligro de que el receptor acepte la misma trama dos o más veces y que la pase a la capa de red más de una vez. Para evitar que esto ocurra, generalmente es necesario asignar números de secuencia a las tramas que salen, a fin de que el receptor pueda distinguir las retransmisiones de los originales.

El asunto de la administración de temporizadores y números de secuencia para asegurar que cada trama llegue finalmente a la capa de red en el destino una sola vez, ni más ni menos, es una parte importante de las tareas de la capa de enlace de datos. Posteriormente en este capítulo estudiaremos la manera en que se lleva a cabo esta administración, observando una serie de ejemplos de complejidad creciente.

3.1.4 Control de flujo

Otro tema de diseño importante que se presenta en la capa de enlace de datos (y también en las capas superiores) es qué hacer con un emisor que quiere transmitir tramas de manera sistemática y a mayor velocidad que aquella con que puede aceptarlos el receptor. Esta situación puede ocurrir fácilmente cuando el emisor opera en una computadora rápida (o con baja carga) y el receptor opera en una máquina lenta (o sobrecargada). El emisor envía las tramas a alta velocidad hasta que satura por completo al receptor. Aunque la transmisión esté libre de errores, en cierto punto el receptor simplemente no será capaz de manejar las tramas conforme lleguen y comenzará a perder algunas. Es obvio que tiene que hacerse algo para evitar esta situación.

Por lo general se utilizan dos métodos. En el primero, el **control de flujo basado en retroalimentación,** el receptor regresa información al emisor autorizándolo para enviar más datos o indicándole su estado. En el segundo, el **control de flujo basado en tasa,** el protocolo tiene un mecanismo integrado que limita la tasa a la que el emisor puede transmitir los datos, sin recurrir a retroalimentación por parte del receptor. En este capítulo estudiaremos el método de control de flujo basado en retroalimentación debido a que el método basado en tasa no se utiliza en la capa de enlace de datos. En el capítulo 5 analizaremos el método basado en tasa.

Se conocen varios esquemas de control de flujo basados en retroalimentación, pero la mayoría se fundamenta en el mismo principio. El protocolo contiene reglas bien definidas respecto al momento en que un emisor puede enviar la siguiente trama. Con frecuencia estas reglas prohíben el envío de tramas hasta que el receptor lo autorice, implícita o explícitamente. Por ejemplo, cuando se establece una conexión, el receptor podría decir: “Puedes enviarme *n* tramas ahora, pero una vez que lo hagas, no envíes nada más hasta que te indique que continúes”. Mas adelante analizaremos los detalles.

3.2 DETECCIÓN Y CORRECCIÓN DE ERRORES

Como vimos en el capítulo 2, el sistema telefónico tiene tres partes: los conmutadores, las troncales interoficinas y los circuitos locales. Las primeras dos son ahora casi enteramente digitales en la mayoría de los países desarrollados. Los circuitos locales aún son cables de par trenzado de cobre analógicos en todos lados y continuarán así durante décadas debido al enorme costo de su reemplazo. Aunque los errores son raros en la parte digital, aún son comunes en los circuitos locales. Además, la comunicación inalámbrica se está volviendo más común, y las tasas de errores son de magnitud mucho mayor que en las troncales de fibra interoficinas. La conclusión es: los errores de transmisión van a ser inevitables durante muchos años más. Tendremos que aprender a lidiar con ellos.

Como resultado de los procesos físicos que los generan, los errores en algunos medios (por ejemplo, la radio) tienden a aparecer en ráfagas y no de manera individual. El hecho de que los errores lleguen en ráfaga tiene ventajas y desventajas con respecto a los errores aislados de un solo bit. Por el lado de las ventajas, los datos de computadora siempre se envían en bloques de bits. Suponga que el tamaño de bloque es de 1000 bits y la tasa de errores es de 0.001 por bit. Si los errores fueran independientes, la mayoría de los bloques contendría un error. Sin embargo, si los errores llegan en ráfagas de 100, en promedio sólo uno o dos bloques de cada 100 serán afectados. La desventaja de los errores en ráfaga es que son mucho más difíciles de detectar y corregir que los errores aislados.

3.2.1 Códigos de corrección de errores

Los diseñadores de redes han desarrollado dos estrategias principales para manejar los errores. Una es incluir suficiente información redundante en cada bloque de datos transmitido para que el receptor pueda deducir lo que debió ser el carácter transmitido. La otra estrategia es incluir sólo suficiente redundancia para permitir que el receptor sepa que ha ocurrido un error (pero no qué error) y entonces solicite una retransmisión. La primera estrategia utiliza **códigos de corrección de errores;** la segunda usa **códigos de detección de errores.** El uso de códigos de corrección de errores usualmente se conoce como **corrección de errores hacia adelante.**

Cada una de estas técnicas ocupa un nicho ecológico diferente. En los canales que son altamente confiables, como los de fibra, es más económico utilizar un código de detección de errores y simplemente retransmitir los bloques defectuosos que surgen ocasionalmente. Sin embargo, en los canales que causan muchos errores, como los enlaces inalámbricos, es mejor agregar la redundancia suficiente a cada bloque para que el receptor pueda descubrir cuál era el bloque original transmitido, en lugar de confiar en una retransmisión, que también podría tener errores.

Para entender la manera en que pueden manejarse los errores, es necesario estudiar de cerca lo que es en realidad un error. Por lo general, una trama consiste en m bits de datos (es decir, de mensaje) y r bits redundantes o de verificación. Sea la longitud total n (es decir, $n = m + r$). A una unidad de n bits que contiene datos y bits de verificación se le conoce como **palabra codificada** de n bits.

Dadas dos palabras codificadas cualesquiera, digamos 10001001 y 10110001, es posible determinar cuántos bits correspondientes difieren. En este caso, difieren tres bits. Para determinar la cantidad de bits diferentes, basta aplicar un OR exclusivo a las dos palabras codificadas y contar la cantidad de bits 1 en el resultado, por ejemplo:

```
10001001
10110001
--------
00111000
```

La cantidad de posiciones de bits en la que difieren dos palabras codificadas se llama **distancia de Hamming** (Hamming, 1950). Su significado es que, si dos palabras codificadas están separadas una distancia de Hamming d, se requerirán d errores de un bit para convertir una en la otra.

En la mayoría de las aplicaciones de transmisión de datos, todos los 2^m mensajes de datos posibles son legales, pero debido a la manera en que se calculan los bits de verificación no se usan todas las 2^n palabras codificadas posibles. Dado el algoritmo de cálculo de los bits de verificación, es posible construir una lista completa de palabras codificadas legales y encontrar, en esta lista, las dos palabras codificadas cuya distancia de Hamming es mínima. Ésta es la distancia de Hamming de todo el código.

Las propiedades de detección y corrección de errores de un código dependen de su distancia de Hamming. Para detectar d errores se necesita un código con distancia $d + 1$, pues con tal código no hay manera de que d errores de un bit puedan cambiar una palabra codificada válida a otra. Cuando el receptor ve una palabra codificada no válida, sabe que ha ocurrido un error de transmisión. De manera similar, para corregir d errores se necesita un código de distancia $2d + 1$, pues así las palabras codificadas legales están tan separadas que, aun con d cambios, la palabra codificada original sigue estando más cercana que cualquier otra palabra codificada, por lo que puede determinarse de manera única.

Como ejemplo sencillo de código de detección de errores, considere un código en el que se agrega un solo **bit de paridad** a los datos. Este bit se escoge de manera que la cantidad de bits 1 en la palabra código sea par (o impar). Por ejemplo, cuando se envía 1011010 con paridad par, se agrega un bit al final, y se vuelve 10110100. Con paridad impar, 1011010 se vuelve 10110101. Un código con un solo bit de paridad tiene una distancia de 2, pues cualquier error de un bit produce una palabra codificada con la paridad equivocada. Este sistema puede usarse para detectar errores individuales.

Como ejemplo sencillo de código de corrección de errores, considere un código con sólo cuatro palabras codificadas válidas:

0000000000, 0000011111, 1111100000 y 1111111111

Este código tiene una distancia de 5, lo que significa que puede corregir errores dobles. Si llega la palabra codificada 0000000111, el receptor sabe que el original debió ser 0000011111. Sin embargo, si un error triple cambia 0000000000 a 0000000111, el error no se corregirá de manera adecuada.

Imagine que deseamos diseñar un código con m bits de mensaje y r bits de verificación que permitirá la corrección de todos los errores individuales. Cada uno de los 2^m mensajes legales tiene n palabras codificadas ilegales a una distancia 1 de él. Éstas se forman invirtiendo en forma sistemática cada uno de los n bits de la palabra codificada de n bits que la forman. Por lo tanto, cada uno de los 2^m mensajes legales requiere $n + 1$ patrones de bits dedicados a él. Dado que la cantidad de patrones de bits es 2^n, debemos tener $(n + 1)2^m \leq 2^n$. Usando $n = m + r$, este requisito se vuelve $(m + r + 1) \leq 2^r$. Dado m, esto impone un límite inferior a la cantidad de bits de verificación necesarios para corregir errores individuales.

De hecho, este límite inferior teórico puede lograrse usando un método gracias a Hamming (1950). Los bits de la palabra codificada se numeran en forma consecutiva, comenzando por el bit 1 a la izquierda, el bit 2 a su derecha inmediata, etcétera. Los bits que son potencias de 2 (1, 2, 4, 8, 16, etcétera) son bits de verificación. El resto (3, 5, 6, 7, 9, etcétera) se rellenan con los m bits de datos. Cada bit de verificación obliga a que la paridad de un grupo de bits, incluyéndolo a él

mismo, sea par (o impar). Un bit puede estar incluido en varios cálculos de paridad. Para ver a qué bits de verificación contribuye el bit de datos en la posición *k,* reescriba *k* como una suma de potencias de 2. Por ejemplo, 11 = 1 + 2 + 8 y 29 = 1 + 4 + 8 + 16. Se comprueba un bit solamente por los bits de verificación que ocurren en su expansión (por ejemplo, el bit 11 es comprobado por los bits 1, 2 y 8).

Cuando llega una palabra codificada, el receptor inicializa a cero un contador y luego examina cada bit de verificación, *k* (*k* = 1, 2, 4, 8,...), para ver si tiene la paridad correcta. Si no, suma *k* al contador. Si el contador es igual a cero tras haber examinado todos los bits de verificación (es decir, si todos fueron correctos), la palabra codificada se acepta como válida. Si el contador es diferente de cero, contiene el número del bit incorrecto. Por ejemplo, si los bits de verificación 1, 2 y 8 tienen errores, el bit invertido es el 11, pues es el único comprobado por los bits 1, 2 y 8. En la figura 3-7 se muestran algunos caracteres ASCII de 7 bits codificados como palabras codificadas de 11 bits usando un código de Hamming. Recuerde que los datos se encuentran en las posiciones de bit 3, 5, 6, 7, 9, 10 y 11.

Carácter	ASCII	Bits de verificación
H	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	01111001111
	0100000	10011000000
c	1100011	11111000011
o	1101111	10101011111
d	1100100	11111001100
e	1100101	00111000101

Orden de transmisión de bits

Figura 3-7. Uso de un código de Hamming para corregir errores en ráfaga.

Los códigos de Hamming sólo pueden corregir errores individuales. Sin embargo, hay un truco que puede servir para que los códigos de Hamming corrijan errores de ráfaga. Se dispone como matriz una secuencia de *k* palabras codificadas consecutivas, con una palabra codificada por fila. Normalmente se transmitiría una palabra codificada a la vez, de izquierda a derecha. Para corregir los errores en ráfaga, los datos deben transmitirse una columna a la vez, comenzando por la columna del extremo izquierdo. Cuando todos los bits *k* han sido enviados, se envía la segunda columna, y así sucesivamente. Cuando la trama llega al receptor, la matriz se reconstruye, una columna a la vez. Si ocurre un error en ráfaga de longitud *k*, cuando mucho se habrá afectado 1 bit de cada una de las *k* palabras codificadas; sin embargo, el código de Hamming puede corregir un error por palabra codificada, así que puede restaurarse la totalidad del bloque. Este método usa *kr* bits de verificación para inmunizar bloques de *km* bits de datos contra un solo error en ráfaga de longitud *k* o menos.

3.2.2 Códigos de detección de errores

Los códigos de corrección de errores se utilizan de manera amplia en los enlaces inalámbricos, que son notoriamente más ruidosos y propensos a errores que el alambre de cobre o la fibra óptica. Sin los códigos de corrección de errores sería difícil pasar cualquier cosa. Sin embargo, a través del cable de cobre o de la fibra óptica, la tasa de error es mucho más baja, por lo que la detección de errores y la retransmisión por lo general son más eficientes ahí para manejar un error ocasional.

Como un ejemplo simple, considere un canal en el que los errores son aislados y la tasa de errores es de 10^{-6} por bit. Sea el tamaño de bloque 1000 bits. Para proporcionar corrección de errores en bloques de 1000 bits se requieren 10 bits de verificación; un megabit de datos requerirá 10000 bits de verificación. Para detectar un solo bloque con 1 bit de error, basta con un bit de paridad por bloque. Por cada 1000 bloques se tendrá que transmitir un bloque extra (1001 bits). La sobrecarga total del método de detección de errores + retransmisión es de sólo 2001 bits por megabit de datos, contra 10,000 bits con un código de Hamming.

Si se agrega un solo bit de paridad a un bloque y el bloque viene muy alterado por una ráfaga de errores prolongada, la probabilidad de que se detecte el error es de 0.5, lo que difícilmente es aceptable. Es posible aumentar la probabilidad considerando a cada bloque por enviar como una matriz rectangular de n bits de ancho y k bits de alto, como se describió anteriormente. Se calcula por separado un bit de paridad para cada columna y se agrega a la matriz como última fila. La matriz se transmite entonces fila por fila. Cuando llega el bloque, el receptor comprueba todos los bits de paridad. Si cualquiera de ellos está mal, solicita la retransmisión del bloque. Se solicitan retransmisiones adicionales hasta que un bloque entero se reciba sin ningún error de paridad.

Este método puede detectar una sola ráfaga de longitud n, pues sólo se cambiará un bit por columna. Sin embargo, una ráfaga de longitud $n + 1$ pasará sin ser detectada si se invierten el primero y último bits, y si todos los demás bits están correctos. (Una ráfaga de errores no implica que todos los bits están mal; sólo implica que cuando menos el primero y el último están mal.) Si el bloque está muy alterado por una ráfaga continua o por múltiples ráfagas más cortas, la probabilidad de que cualquiera de las n columnas tenga, por accidente, la paridad correcta es de 0.5, por lo que la probabilidad de aceptar un bloque alterado cuando no se debe es de 2^{-n}.

Aunque en algunos casos el método anterior puede ser adecuado, en la práctica se usa uno muy definido: el **código polinomial** (también conocido como **código de redundancia cíclica** o **código CRC**). Los códigos polinomiales se basan en el tratamiento de cadenas de bits como representaciones de polinomios con coeficientes de 0 y 1 solamente. Una trama de k bits se considera como la lista de coeficientes de un polinomio con k términos que van de x^{k-1} a x^0. Se dice que tal polinomio es de grado $k - 1$. El bit de orden mayor (que se encuentra más a la izquierda) es el coeficiente de x^{k-1}, el siguiente bit es el coeficiente de x^{k-2} y así sucesivamente. Por ejemplo, 110001 tiene 6 bits y, por lo tanto, representa un polinomio de seis términos con coeficientes 1, 1, 0, 0, 0 y 1: $x^5 + x^4 + x^0$.

La aritmética polinomial se hace mediante una operación módulo 2, de acuerdo con las reglas de la teoría de campos algebraicos. No hay acarreos para la suma, ni préstamos para la resta. Tanto la suma como la resta son idénticas a un OR exclusivo. Por ejemplo:

10011011	00110011	11110000	01010101
+11001010	+11001101	−10100110	−10101111
01010001	11111110	01010110	11111010

La división se lleva a cabo de la misma manera que en binario, excepto que la resta es módulo 2, igual que antes. Se dice que un divisor "cabe" en un dividendo si éste tiene tantos bits como el divisor.

Cuando se emplea el método de código polinomial, el emisor y el receptor deben acordar por adelantado un **polinomio generador,** $G(x)$. Tanto los bits de orden mayor y menor del generador deben ser 1. Para calcular la **suma de verificación** para una trama con m bits, correspondiente al polinomio $M(x)$, la trama debe ser más larga que el polinomio generador. La idea es incluir una suma de verificación al final de la trama de tal manera que el polinomio representado por la trama con suma de verificación sea divisible entre $G(x)$. Cuando el receptor recibe la trama con suma de verificación, intenta dividirla entre $G(x)$. Si hay un residuo, ha habido un error de transmisión.

El algoritmo para calcular la suma de verificación es el siguiente:

1. Sea r el grado de $G(x)$. Anexe r bits cero al final de la trama, para que ahora contenga $m + r$ bits y corresponda al polinomio $x^r M(x)$.
2. Divida la cadena de bits correspondiente a $G(x)$ entre la correspondiente a $x^r M(x)$ usando una división módulo 2.
3. Reste el residuo (que siempre es de r o menos bits) a la cadena de bits correspondiente a $x^r M(x)$ usando una resta módulo 2. El resultado es la trama con suma de verificación que va a transmitirse. Llame a su polinomio $T(x)$.

En la figura 3-8 se ilustra el cálculo para una trama 1101011011 utilizando el generador $G(x) = x^4 + x + 1$.

Debe quedar claro que $T(x)$ es divisible (módulo 2) entre $G(x)$. En cualquier problema de división, si se resta el residuo del dividendo, lo que queda es divisible entre el divisor. Por ejemplo, en base 10, si se divide 210,278 entre 10,941, el residuo es 2399. Si se resta 2399 a 210,278, lo que queda (207,879) es divisible entre 10,941.

Ahora analizaremos el alcance de este método. ¿Qué tipos de error se detectarán? Imagine que ocurre un error de transmisión tal que en lugar de que llegue la cadena de bits para $T(x)$, llega $T(x) + E(x)$. Cada bit 1 en $E(x)$ corresponde a un bit que ha sido invertido. Si hay k bits 1 en $E(x)$, han ocurrido k errores de un solo bit. Una ráfaga de errores individual se caracteriza por un 1 inicial, una mezcla de ceros y unos, y un 1 final, siendo los demás bits 0.

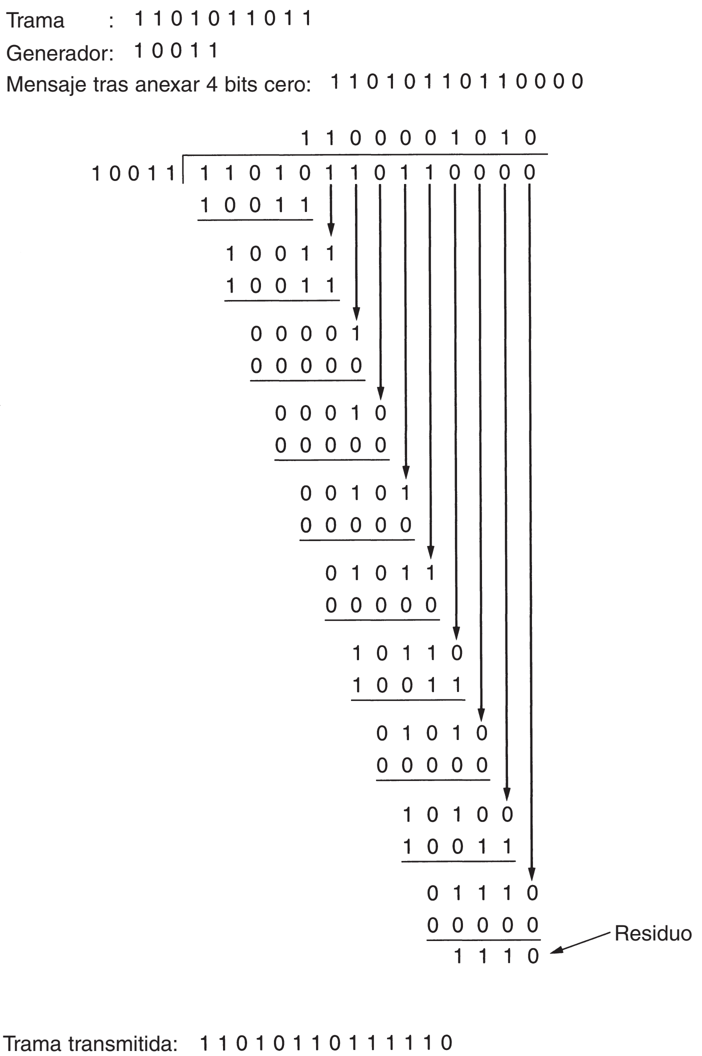


Figura 3-8. Cálculo de la suma de verificación de código polinomial.

Al recibir la trama con suma de verificación, el receptor la divide entre $G(x)$; es decir, calcula $[T(x) + E(x)]/G(x)$. $T(x)/G(x)$ es 0, por lo que el resultado del cálculo es simplemente $E(x)/G(x)$. No se detectarán los errores que por casualidad correspondan a polinomios que contengan $G(x)$ como factor; todos los demás errores serán atrapados.

Si ha ocurrido un error de un solo bit, $E(x) = x^i$, donde i determina qué bit es erróneo. Si $G(x)$ contiene dos o más términos, nunca será divisor exacto de $E(x)$, por lo que se detectarán los errores de un solo bit.

Si han ocurrido dos errores de un solo bit aislados, $E(x) = x^i + x^j$, donde $i > j$. Esto también se puede escribir como $E(x) = x^j(x^{i-j} + 1)$. Si suponemos que $G(x)$ no es divisible entre x, una condición suficiente para detectar todos los errores dobles es que $G(x)$ no divida a $x^k + 1$ para ninguna k hasta el valor máximo de $i - j$ (es decir, hasta la longitud máxima de la trama). Se conocen polinomios sencillos de bajo grado que dan protección a tramas largas. Por ejemplo, $x^{15} + x^{14} + 1$ no será divisor exacto de $x^k + 1$ para ningún valor de k menor que 32,768.

Si hay una cantidad impar de bits con error, $E(x)$ contiene un número impar de términos (por ejemplo, $x^5 + x^2 + 1$, pero no $x^2 + 1$). Curiosamente, ningún polinomio con un número impar de términos posee a $x + 1$ como un factor en el sistema de módulo 2. Haciendo $x + 1$ un factor de $G(x)$, podemos atrapar todos los errores consistentes en un número impar de bits invertidos.

Para comprobar que ningún polinomio con una cantidad impar de términos es divisible entre $x + 1$, suponga que $E(x)$ tiene un número impar de términos y que es divisible entre $x + 1$. Factorice $E(x)$ en $(x + 1)\,Q(x)$. Ahora evalúe $E(1) = (1 + 1)Q(1)$. Dado que $1 + 1 = 0$ (módulo 2), $E(1)$ debe ser cero. Si $E(x)$ tiene un número impar de términos, la sustitución de 1 por x en cualquier lugar siempre dará como resultado un 1. Por lo tanto, ningún polinomio con un número impar de términos es divisible entre $x + 1$.

Por último, y lo que es más importante, un código polinomial con r bits de verificación detectará todos los errores en ráfaga de longitud $\leq r$. Un error en ráfaga de longitud k puede representarse mediante $x^i(x^{k-1} +. . .+ 1)$, donde i determina la distancia a la que se encuentra la ráfaga desde el extremo derecho de la trama recibida. Si $G(x)$ contiene un término x^0, no tendrá x^i como factor, por lo que, si el grado de la expresión entre paréntesis es menor que el grado de $G(x)$, el residuo nunca puede ser cero.

Si la longitud de la ráfaga es de $r + 1$, el residuo de la división entre $G(x)$ será cero si, y sólo si, la ráfaga es idéntica a $G(x)$. Por la definición de ráfaga, el primero y el último bit deben ser 1, así que el que sean iguales o no depende de los $r - 1$ bits intermedios. Si se consideran igualmente probables todas las combinaciones, la probabilidad de que se acepte como válida tal trama incorrecta es de ${}^1\!/\!{}_2{}^{r-1}$.

También puede demostrarse que cuando ocurre una ráfaga de errores mayor que $r + 1$, o cuando ocurren varias ráfagas más cortas, la probabilidad de que una trama incorrecta no sea detectada es de ${}^1\!/\!{}_2{}^{r}$, suponiendo que todos los patrones de bits sean igualmente probables.

Ciertos polinomios se han vuelto estándares internacionales. El que se utiliza en el IEEE 802 es:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$$

Entre otras propiedades deseables, tiene la de que detecta todas las ráfagas con una longitud de 32 o menor y todas las ráfagas que afecten un número impar de bits.

Aunque el cálculo requerido para obtener la suma de verificación puede parecer complicado, Peterson y Brown (1961) han demostrado que puede construirse un circuito sencillo con un registro de desplazamiento para calcular y comprobar las sumas de verificación por hardware. En la práctica, casi siempre se usa este hardware. La mayoría de las LANs lo utiliza y, en algunos casos, también lo hacen las líneas punto a punto.

Durante décadas se ha supuesto que las tramas para las que se generan sumas de verificación contienen bits aleatorios. Todos los análisis de algoritmos de suma de verificación se han hecho bajo este supuesto. En fechas más recientes, la inspección de datos reales ha demostrado que este supuesto es equivocado. Como consecuencia, en algunas circunstancias los errores no detectados son mucho más comunes de lo que se pensaba anteriormente (Partridge y cols., 1995).

3.3 PROTOCOLOS ELEMENTALES DE ENLACE DE DATOS

Como introducción al tema de los protocolos, comenzaremos por estudiar tres protocolos de complejidad creciente. Los lectores interesados pueden conseguir un simulador de estos protocolos y otros subsecuentes a través de WWW (vea el prefacio). Antes de estudiar los protocolos, es útil hacer explícitos algunos de los supuestos implícitos del modelo de comunicaciones. Para comenzar, estamos suponiendo que en las capas física, de enlace de datos y de red hay procesos independientes que se comunican pasando mensajes de un lado a otro. En muchos casos, los procesos de las capas física y de enlace de datos se ejecutan en un procesador dentro de un chip especial de E/S y los de la capa de red lo hacen en la CPU principal. Sin embargo, también puede haber otras implementaciones (por ejemplo, tres procesos en un solo chip de E/S o las capas física y de enlace de datos como procedimientos invocados por el proceso de la capa de red). En cualquier caso, el hecho de tratar las tres capas como procesos independientes hace más nítido el análisis en el terreno conceptual y también sirve para subrayar la independencia de las capas.

Otro supuesto clave es que la máquina *A* desea mandar un flujo considerable de datos a la máquina *B* usando un servicio confiable orientado a la conexión. Después consideraremos el caso en que *B* también quiere mandar datos a *A* de manera simultánea. Se ha supuesto que *A* tiene un suministro infinito de datos listos para ser enviados y nunca tiene que esperar a que se produzcan datos. Cuando la capa de enlace de datos de *A* solicita datos, la capa de red siempre es capaz de proporcionarlos de inmediato. (Esta restricción también se desechará posteriormente.)

También supondremos que las máquinas no fallan. Es decir, estos protocolos manejan errores de comunicación, pero no los problemas causados por computadoras que fallan y se reinician.

En lo que concierne a la capa de enlace de datos, el paquete que se le pasa a través de la interfaz desde la capa de red es de datos puros, que deben ser entregados bit por bit a la capa de red del destino. El hecho de que la capa de red del destino pueda interpretar parte del paquete como un encabezado no es de importancia para la capa de enlace de datos.

Cuando la capa de enlace de datos acepta un paquete, lo encapsula en una trama agregándole un encabezado y un terminador de enlace de datos (vea la figura 3-1). Por lo tanto, una trama consiste en un paquete incorporado, cierta información de control (en el encabezado) y una suma de verificación (en el terminador). A continuación la trama se transmite a la capa de enlace de datos de la otra máquina. Supondremos que existen procedimientos de biblioteca adecuados *to_physical_layer* para enviar una trama y *from_physical_layer* para recibir una trama. El hardware emisor calcula y agrega la suma de verificación (y de esta manera crea el terminador) por lo que el software

de la capa de enlace de datos no necesita preocuparse por ella. Por ejemplo, podría utilizarse el algoritmo polinomial analizado antes en este capítulo.

Inicialmente el receptor no tiene nada que hacer. Sólo está esperando que ocurra algo. En los protocolos de ejemplo de este capítulo indicamos que la capa de enlace de datos está en espera de que ocurra algo con la llamada de procedimiento *wait_for_event*(*&event*). Este procedimiento sólo regresa cuando ocurre algo (por ejemplo, cuando llega una trama). Al regresar, la variable *event* indica lo que ha ocurrido. El grupo de eventos posibles difiere para cada uno de los diferentes protocolos que describiremos, y se definirán por separado para cada protocolo. Observe que en una situación más realista, la capa de enlace de datos no se quedará en un ciclo cerrado esperando un evento, como hemos sugerido, sino que recibirá una interrupción, la que ocasionará que suspenda lo que estaba haciendo y proceda a manejar la trama entrante. Sin embargo, por sencillez ignoraremos todos los detalles de la actividad paralela en la capa de enlace de datos y daremos por hecho que la capa está dedicada de tiempo completo a manejar nuestro canal.

Cuando llega una trama al receptor, el hardware calcula la suma de verificación. Si ésta es incorrecta (es decir, si hubo un error de transmisión), se le informa a la capa de enlace de datos (*event* = *cksum_err*). Si la trama entrante llega sin daño, también se le informa a la capa de enlace de datos (*event* = *frame_arrival*) para que pueda adquirir la trama para inspeccionarla usando *from_physical_layer*. Tan pronto como la capa de enlace de datos receptora adquiere una trama sin daños, revisa la información de control del encabezado y, si todo está bien, pasa la parte que corresponde al paquete a la capa de red. En ninguna circunstancia se entrega un encabezado de trama a la capa de red.

Hay una buena razón por la que la capa de red nunca debe recibir ninguna parte del encabezado de trama: para mantener completamente separados el protocolo de red y el de enlace de datos. En tanto la capa de red no sepa nada en absoluto sobre el protocolo de enlace de datos ni el formato de la trama, éstos podrán cambiarse sin requerir cambios en el software de la capa de red. Al proporcionarse una interfaz rígida entre la capa de red y la de enlace de datos se simplifica en gran medida el diseño del software, pues los protocolos de comunicación de las diferentes capas pueden evolucionar en forma independiente.

En la figura 3-9 se muestran algunas declaraciones comunes (en C) para muchos de los protocolos que se analizarán después. Allí se definen cinco estructuras de datos: *boolean*, *seq_nr*, *packet*, *frame_kind* y *frame*. Un *boolean* es un tipo de dato numérico que puede tener los valores *true* y *false*. Un *seq_nr* (número de secuencia) es un entero pequeño que sirve para numerar las tramas, a fin de distinguirlas. Estos números de secuencia van de 0 hasta *MAX_SEQ* (inclusive), que se define en cada protocolo que lo necesita. Un *packet* es la unidad de intercambio de información entre la capa de red y la de enlace de datos en la misma máquina, o entre entidades iguales de la capa de red. En nuestro modelo siempre contiene *MAX_PKT* bytes, pero en la práctica sería de longitud variable.

Un *frame* está compuesto de cuatro campos: *kind, Seq*, *ack* e *info*. Los primeros tres contienen información de control y el último puede contener los datos por transferir. Estos campos de control constituyen en conjunto el **encabezado de la trama.**

```
#define MAX_PKT 1024                          /* determina el tamaño del paquete
                                                 en bytes */

typedef enum {false, true} boolean;           /* tipo booleano */
typedef unsigned int seq_nr;                  /* números de secuencia o
                                                 confirmación */
typedef struct {unsigned char data[MAX_PKT];} packet;/*  definición de paquete */
typedef enum {data, ack, nak} frame_kind;     /* definición de frame_kind */

typedef struct {                              /* las tramas se transportan en
                                                 esta capa */
 frame_kind kind;                             /* ¿qué clase de trama es? */
 seq_nr seq;                                  /* número de secuencia */
 seq_nr ack;                                  /* número de confirmación de
                                                 recepción */
 packet info;                                 /* paquete de la capa de red */
} frame;

/* Espera que ocurra un evento; devuelve el tipo en la variable event. */
void wait_for_event(event_type *event);

/* Obtiene un paquete de la capa de red para transmitirlo por el canal. */
void from_network_layer(packet *p);

/* Entrega información de una trama entrante a la capa de red. */
void to_network_layer(packet *p);

/* Obtiene una trama entrante de la capa física y la copia en r. */
void from_physical_layer(frame *r);

/* Pasa la trama a la capa física para transmitirla. */
void to_physical_layer(frame *s);

/* Arranca el reloj y habilita el evento de expiración de temporizador. */
void start_timer(seq_nr k);

/* Detiene el reloj e inhabilita el evento de expiración de temporizador. */
void stop_timer(seq_nr k);

/* Inicia un temporizador auxiliar y habilita el evento ack_timeout. */
void start_ack_timer(void);

/* Detiene el temporizador auxiliar e inhabilita el evento ack_timeout. */
void stop_ack_timer(void);

/* Permite que la capa de red cause un evento network_layer_ready. */
void enable_network_layer(void);

/* Evita que la capa de red cause un evento network_layer_ready. */
void disable_network_layer(void);

/* La macro inc se expande en línea: incrementa circularmente k. */
#define inc(k) if (k < MAX_SEQ) k = k + 1; else k = 0
```

Figura 3-9. Algunas definiciones necesarias en los protocolos que siguen. Estas definiciones se encuentran en el archivo *protocol.h.*

El campo *kind* indica si hay datos en la trama, porque algunos de los protocolos distinguen entre las tramas que contienen exclusivamente información de control y los que también contienen datos. Los campos *seq* y *ack* se emplean para números de secuencia y confirmaciones de recepción, respectivamente; su uso se describirá posteriormente con mayor detalle. El campo *info* de una trama de datos contiene un solo paquete; el campo *info* de una trama de control no se usa. En una implementación más realista se usaría un campo *info* de longitud variable, omitiéndolo por completo en las tramas de control.

Es importante entender la relación entre un paquete y una trama. La capa de red construye un paquete tomando un mensaje de la capa de transporte y agregándole el encabezado de la capa de red. Este paquete se pasa a la capa de enlace de datos para incluirlo en el campo *info* de una trama saliente. Cuando ésta llega a su destino, la capa de enlace de datos extrae de ella el paquete y a continuación lo pasa a la capa de red. De esta manera, esta capa puede actuar como si las máquinas pudieran intercambiar paquetes directamente.

En la figura 3-9 también se listan varios procedimientos que son rutinas de biblioteca cuyos detalles dependen de la implementación, por lo que no nos ocuparemos de su funcionamiento interno aquí. El procedimiento *wait_ for_event* se queda en un ciclo cerrado esperando que algo ocurra, como se mencionó antes. Con los procedimientos *to_network_layer* y *from_network_layer,* la capa de enlace de datos pasa paquetes a la capa de red y acepta paquetes de ella, respectivamente. Observe que *from_physical_layer* y *to_physical_layer* pasan tramas entre la capa de enlace de datos y la capa física, y que los procedimientos *to_network_layer* y *from_network_layer* pasan paquetes entre la capa de enlace de datos y la capa de red. En otras palabras, *to_network_layer* y *from_network_layer* tienen que ver con la interfaz entre las capas 2 y 3, y *from_physical_layer* y *to_physical_layer,* con la interfaz entre las capas 1 y 2.

En la mayoría de los protocolos suponemos un canal inestable que pierde tramas completas ocasionalmente. Para poder recuperarse de tales calamidades, la capa de enlace de datos emisora debe arrancar un temporizador o reloj interno cada vez que envía una trama. Si no obtiene respuesta tras transcurrir cierto intervalo de tiempo predeterminado, el temporizador expira y la capa de enlace de datos recibe una señal de interrupción.

En nuestros protocolos, esto se maneja permitiendo que el procedimiento *wait_ for_event* devuelva *event* = *timeout*. Los procedimientos *start_timer* y *stop_timer* inician y detienen, respectivamente, el temporizador. Las terminaciones del temporizador sólo son posibles cuando éste se encuentra en funcionamiento. Se permite explícitamente llamar a *start_timer* cuando el temporizador está funcionando; tal llamada tan sólo restablece el reloj para hacer que el temporizador termine después de haber transcurrido un intervalo completo de temporización (a menos que se restablezca o apague antes).

Los procedimientos *star_ack_timer* y *stop_ack_timer* controlan un temporizador auxiliar usado para generar confirmaciones de recepción en ciertas condiciones.

Los procedimientos *enable_network_layer* y *disable_network_layer* se usan en los protocolos más complicados, en los que ya no suponemos que la capa de red siempre tiene paquetes que enviar. Cuando la capa de enlace de datos habilita a la capa de red, ésta tiene permitido interrumpir cuando tenga que enviar un paquete. Esto lo indicamos con *event* = *network_layer_ready*. Cuando una capa de red está inhabilitada, no puede causar tales eventos. Teniendo cuidado respecto a

cuándo habilitar e inhabilitar su capa de red, la capa de enlace de datos puede evitar que la capa de red la sature con paquetes para los que no tiene espacio de búfer.

Los números de secuencia de las tramas siempre están en el intervalo de 0 a *MAX_SEQ* (inclusive), donde *MAX_SEQ* es diferente para los distintos protocolos. Con frecuencia es necesario avanzar circularmente en 1 un número de secuencia (por ejemplo, *MAX_SEQ* va seguido de 0). La macro *inc* lleva a cabo este incremento. Esta función se ha definido como macro porque se usa en línea dentro de la ruta crítica. Como veremos después en este libro, el factor que limita el desempeño de una red con frecuencia es el procesamiento del protocolo, por lo que definir como macros las operaciones sencillas como ésta no afecta la legibilidad del código y sí mejora el desempeño. Además, ya que *MAX_SEQ* tendrá diferentes valores en diferentes protocolos, al hacer que *inc* sea una macro, cabe la posibilidad de incluir todos los protocolos en el mismo binario sin conflictos. Esta capacidad es útil para el simulador.

Las declaraciones de la figura 3-9 son parte de todos los protocolos que siguen. Para ahorrar espacio y proporcionar una referencia práctica, se han extraído y listado juntas, pero conceptualmente deberían estar integradas con los protocolos mismos. En C, esta integración se efectúa poniendo las definiciones en un archivo especial de encabezado, en este caso *protocol.h*, y usando la directiva #include del preprocesador de C para incluirlas en los archivos de protocolos.

3.3.1 Un protocolo símplex sin restricciones

Como ejemplo inicial consideraremos un protocolo que es lo más sencillo posible. Los datos se transmiten sólo en una dirección; las capas de red tanto del emisor como del receptor siempre están listas; el tiempo de procesamiento puede ignorarse; hay un espacio infinito de búfer y, lo mejor de todo, el canal de comunicación entre las capas de enlace de datos nunca tiene problemas ni pierde tramas. Este protocolo completamente irreal, al que apodaremos “utopía”, se muestra en la figura 3-10.

El protocolo consiste en dos procedimientos diferentes, uno emisor y uno receptor. El emisor se ejecuta en la capa de enlace de datos de la máquina de origen y el receptor se ejecuta en la capa de enlace de datos de la máquina de destino. No se usan números de secuencia ni confirmaciones de recepción, por lo que no se necesita *MAX_SEQ*. El único tipo de evento posible es *frame_arrival* (es decir, la llegada de una trama sin daños).

El emisor está en un ciclo `while` infinito que sólo envía datos a la línea tan rápidamente como puede. El cuerpo del ciclo consiste en tres acciones: obtener un paquete de la (siempre dispuesta) capa de red, construir una trama de salida usando la variable *s* y enviar la trama a su destino. Este protocolo sólo utiliza el campo *info* de la trama, pues los demás campos tienen que ver con el control de errores y de flujo, y aquí no hay restricciones de control de errores ni de flujo.

El receptor también es sencillo. Inicialmente, espera que algo ocurra, siendo la única posibilidad la llegada de una trama sin daños. En algún momento, la trama llega y el procedimiento *wait_for_event* regresa, conteniendo *event* el valor *frame_arrival* (que de todos modos se ignora). La llamada a *from_physical_layer* elimina la trama recién llegada del búfer de hardware y la

```
/* El protocolo 1 (utopía) provee la transmisión de datos en una sola
dirección, del emisor al  receptor. Se supone que el canal de comunicación
está libre de errores, y que el receptor es capaz de procesar toda la entrada
a una rapidez infinita. En consecuencia, el emisor se  mantiene en un ciclo,
enviando datos a la línea tan rápidamente como puede. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender1(void)
{
  frame s;                          /* búfer para una trama de salida */
  packet buffer;                    /* búfer para un paquete de salida */

  while (true) {
     from_network_layer(&buffer); /* consigue algo que enviar */
     s.info = buffer;              /* lo copia en s para transmisión */
     to_physical_layer(&s);        /* lo envía a su destino */
  }                                 /* Mañana, y mañana, y mañana,
                                      Se arrastra a este mísero paso de día
                                        a día
                                      Hasta la última sílaba del tiempo
                                        recordado
                                          -Macbeth, V, v */

}

void receiver1(void)
{
  frame r;
  event_type event;                 /* ocupado por wait, pero no se usa aquí */

  while (true) {
     wait_for_event(&event);       /* la única posibilidad es frame_arrival */
     from_physical_layer(&r);      /* obtiene la trama entrante */
     to_network_layer(&r.info);    /* pasa los datos a la capa de red */
  }
}
```

Figura 3-10. Protocolo símplex sin restricciones.

coloca en la variable *r*, en donde el código receptor pueda obtenerla. Por último, la parte de datos se pasa a la capa de red y la capa de enlace de datos se retira para esperar la siguiente trama, suspendiéndose efectivamente hasta que llega la trama.

3.3.2 Protocolo símplex de parada y espera

Ahora omitiremos el supuesto más irreal hecho en el protocolo 1: la capacidad de la capa de red receptora de procesar datos de entrada con una rapidez infinita (o, lo que es equivalente, la presencia en la capa de enlace de datos receptora de una cantidad infinita de espacio de búfer en el cual almacenar todas las tramas de entrada mientras esperan su respectivo turno). Todavía se supone que el canal de comunicaciones está libre de errores y que el tráfico de datos es símplex.

El problema principal que debemos resolver aquí es cómo evitar que el emisor sature al receptor enviando datos a mayor velocidad de la que este último puede procesarlos. En esencia, si el receptor requiere un tiempo Δt para ejecutar *from_physical_layer* más *to_network_layer*, el emisor debe transmitir a una tasa media menor que una trama por tiempo Δt. Es más, si suponemos que en el hardware del receptor no se realiza de manera automática el almacenamiento en el búfer y el encolamiento, el emisor nunca debe transmitir una trama nueva hasta que la vieja haya sido obtenida por *from_ physical_layer*, para que lo nuevo no sobrescriba lo antiguo.

En ciertas circunstancias restringidas (por ejemplo, transmisión síncrona y una capa de enlace de datos receptora dedicada por completo a procesar la línea de entrada única), el emisor podría introducir simplemente un retardo en el protocolo 1 y así reducir su velocidad lo suficiente para evitar que se sature el receptor. Sin embargo, es más común que la capa de enlace de datos tenga varias líneas a las cuales atender, y el intervalo de tiempo entre la llegada de una trama y su procesamiento puede variar en forma considerable. Si los diseñadores de la red pueden calcular el comportamiento de peor caso del receptor, podrán programar al emisor para que transmita con tanta lentitud que, aun si cada trama sufre el retardo máximo, no haya desbordamientos. El problema con este método es que es demasiado conservador. Conduce a un aprovechamiento del ancho de banda muy por debajo del óptimo, a menos que el mejor caso y el peor sean iguales (es decir, la variación en el tiempo de reacción de la capa de enlace de datos sea pequeña).

Una solución más general para este dilema es hacer que el receptor proporcione retroalimentación al emisor. Tras haber pasado un paquete a su capa de red, el receptor regresa al emisor una pequeña trama ficticia que, de hecho, autoriza al emisor para transmitir la siguiente trama. Tras haber enviado una trama, el protocolo exige que el emisor espere hasta que llegue la pequeña trama ficticia (es decir, la confirmación de recepción). Utilizar la retroalimentación del receptor para indicar al emisor cuándo puede enviar más datos es un ejemplo del control de flujo que se mencionó anteriormente.

Los protocolos en los que el emisor envía una trama y luego espera una confirmación de recepción antes de continuar se denominan de **parada y espera.** En la figura 3-11 se da un ejemplo de un protocolo símplex de parada y espera.

Aunque el tráfico de datos en este ejemplo es símplex, y va sólo desde el emisor al receptor, las tramas viajan en ambas direcciones. En consecuencia, el canal de comunicación entre las dos capas de enlace de datos necesita tener capacidad de transferencia de información bidireccional. Sin embargo, este protocolo implica una alternancia estricta de flujo: primero el emisor envía una trama, después el receptor envía una trama, después el emisor envía otra trama, después el receptor envía otra, y así sucesivamente. Aquí sería suficiente un canal físico semidúplex.

```
/* El protocolo 2 (parada y espera) también contempla un flujo unidireccional
de datos del emisor al receptor. Se da por hecho nuevamente que el canal de
comunicación está libre de errores, como en el protocolo 1. Sin embargo, esta
vez el receptor tiene capacidad finita de búfer y capacidad finita de
procesamiento, por lo que el protocolo debe evitar de manera explícita que
el emisor sature al receptor con datos a mayor velocidad de la que puede
manejar. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender2(void)
{
  frame s;                          /* búfer para una trama de salida */
  packet buffer;                    /* búfer para un paquete de salida */
  event_type event;                 /* frame_arrival es la única posibilidad */

  while (true) {
     from_network_layer(&buffer);   /* consigue algo que enviar */
     s.info = buffer;               /* lo copia en s para transmisión */
     to_physical_layer(&s);         /* adiós a la pequeña trama */
     wait_for_event(&event);        /* no procede hasta que recibe la señal de
                                         continuación */
  }
}

void receiver2(void)
{
  frame r, s;                       /* búferes para las tramas */
  event_type event;                 /* frame_arrival es la única posibilidad */
  while (true) {
     wait_for_event(&event);        /* la única posibilidad es frame_arrival */
     from_physical_layer(&r);       /* obtiene la trama entrante */
     to_network_layer(&r.info);     /* pasa los datos a la capa de red */
     to_physical_layer(&s);         /* envía una trama ficticia para informar
                                       al emisor */
  }
}
```

Figura 3-11. Protocolo símplex de parada y espera.

Al igual que en el protocolo 1, el emisor comienza obteniendo un paquete de la capa de red, usándolo para construir una trama y enviarla a su destino. Sólo que ahora, a diferencia del protocolo 1, el emisor debe esperar hasta que llegue una trama de confirmación de recepción antes de reiniciar el ciclo y obtener el siguiente paquete de la capa de red. La capa de enlace de datos emisora no necesita inspeccionar la trama entrante, ya que sólo hay una posibilidad: la trama siempre es de confirmación de recepción.

La única diferencia entre *receiver1* y *receiver2* es que, tras entregar un paquete a la capa de red, *receiver2* regresa al emisor una trama de confirmación de recepción antes de entrar nuevamente en el ciclo de espera. Puesto que sólo es importante la llegada de la trama en el emisor, no su contenido, el receptor no necesita poner ninguna información específica en él.

3.3.3 Protocolo símplex para un canal con ruido

Ahora consideremos la situación normal de un canal de comunicación que comete errores. Las tramas pueden llegar dañadas o perderse por completo. Sin embargo, suponemos que si una trama se daña en tránsito, el hardware del receptor detectará esto cuando calcule la suma de verificación. Si la trama está dañada de tal manera que pese a ello la suma de verificación sea correcta, un caso excesivamente improbable, este protocolo (y todos los demás) puede fallar (es decir, entregar un paquete incorrecto a la capa de red).

A primera vista puede parecer que funcionaría una variación del protocolo 2: agregar un temporizador. El emisor podría enviar una trama, pero el receptor sólo enviaría una trama de confirmación de recepción si los datos llegaran correctamente. Si llegara una trama dañada al receptor, se desecharía. Poco después, el temporizador del emisor expiraría y se enviaría la trama de nuevo. Este proceso se repetiría hasta que la trama por fin llegara intacta.

El esquema anterior tiene un defecto mortal. Medite el problema e intente descubrir lo que podría fallar antes de continuar leyendo.

Para ver lo que puede resultar mal, recuerde que la capa de enlace de datos debe proporcionar una comunicación transparente y libre de errores entre los procesos de las capas de red. La capa de red de la máquina *A* pasa una serie de paquetes a la capa de enlace de datos, que debe asegurar que se entregue una serie de paquetes idéntica a la capa de red de la máquina *B* a través de su capa de enlace de datos. En particular, la capa de red en *B* no tiene manera de saber si el paquete se ha perdido o se ha duplicado, por lo que la capa de enlace de datos debe garantizar que ninguna combinación de errores de transmisión, por improbables que sean, pueda causar la entrega de un paquete duplicado a la capa de red.

Considere el siguiente escenario:

1. La capa de red de *A* entrega el paquete 1 a su capa de enlace de datos. El paquete se recibe correctamente en *B* y se pasa a la capa de red de *B*. *B* regresa a *A* una trama de confirmación de recepción.

2. La trama de confirmación de recepción se pierde por completo. Nunca llega. La vida sería mucho más sencilla si el canal sólo alterara o perdiera tramas de datos y no tramas de control, pero desgraciadamente el canal no hace distinciones.

3. El temporizador de la capa de enlace de datos de *A* expira en algún momento. Al no haber recibido una confirmación de recepción, supone (incorrectamente) que su trama de datos se ha perdido o dañado, y envía otra vez la trama que contiene el paquete 1.

4. La trama duplicada también llega bien a la capa de enlace de datos de *B* y de ahí se pasa de manera inadvertida a la capa de red. Si *A* está enviando un archivo a *B*, parte del archivo se duplicará (es decir, la copia del archivo reconstruida por *B* será incorrecta y el error no se habrá detectado). En otras palabras, el protocolo fallará.

Es claro que lo que se necesita es alguna manera de que el receptor sea capaz de distinguir entre una trama que está viendo por primera vez y una retransmisión. La forma evidente de lograr esto es hacer que el emisor ponga un número de secuencia en el encabezado de cada trama que envía. A continuación, el receptor puede examinar el número de secuencia de cada trama que llega para ver si es una trama nueva o un duplicado que debe descartarse.

Dado que es deseable que el encabezado de las tramas sea pequeño, surge la pregunta: ¿cuál es la cantidad mínima de bits necesarios para el número de secuencia? La única ambigüedad de este protocolo es entre una trama, m, y su sucesor directo, $m + 1$. Si la trama m se pierde o se daña, el receptor no confirmará su recepción y el emisor seguirá tratando de enviarla. Una vez que la trama se recibe correctamente, el receptor regresa una confirmación de recepción al emisor. Es aquí donde surge el problema potencial. Dependiendo de si el emisor recibe correctamente la trama de confirmación de recepción, tratará de enviar m o $m + 1$.

El evento que indica al emisor que puede enviar $m + 2$ es la llegada de una confirmación de recepción de $m + 1$. Pero esto implica que m se recibió de manera correcta, y además que su confirmación de recepción fue recibida correctamente por el emisor (de otra manera, el emisor no habría comenzado con $m + 1$, y mucho menos con $m + 2$). Como consecuencia, la única ambigüedad es entre una trama y su antecesor o sucesor inmediatos, no entre el antecesor y el sucesor mismos.

Por lo tanto, basta con un número de secuencia de 1 bit (0 o 1). En cada instante, el receptor espera un número de secuencia en particular. Cualquier trama de entrada que contenga un número de secuencia equivocado se rechaza como duplicado. Cuando llega una trama que contiene el número de secuencia correcto, se acepta y se pasa a la capa de red, y el número de secuencia esperado se incrementa módulo 2 (es decir, 0 se vuelve 1 y 1 se vuelve 0).

En la figura 3-12 se muestra un ejemplo de este tipo de protocolo. Los protocolos en los que el emisor espera una confirmación de recepción positiva antes de avanzar al siguiente elemento de datos suelen llamarse **PAR (Confirmación de Recepción Positiva con Retransmisión)** o **ARQ (Solicitud Automática de Repetición).** Al igual que el protocolo 2, éste también transmite datos en una sola dirección.

El protocolo 3 difiere de sus antecesores en que tanto el emisor como el receptor tienen una variable cuyo valor se recuerda mientras la capa de enlace de datos está en estado de espera. El emisor recuerda el número de secuencia de la siguiente trama a enviar en *next_frame_to_send*; el receptor recuerda el número de secuencia de la siguiente trama esperada en *frame_expected*. Cada protocolo tiene una fase de inicialización corta antes de entrar en el ciclo infinito.

```
/* El protocolo 3 (par) permite el flujo unidireccional de datos por un canal no con-
fiable. */

#define MAX_SEQ 1                              /* debe ser 1 para el protocolo 3 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void sender3(void)
{
  seq_nr next_frame_to_send;                   /* número de secuencia de la siguiente
                                                  trama de salida */
  frame s;                                     /* variable de trabajo */
  packet buffer;                               /* búfer para un paquete de salida */
  event_type event;

  next_frame_to_send = 0;                      /* inicializa números de secuencia de
                                                  salida */
  from_network_layer(&buffer);                 /* obtiene el primer paquete */
  while (true){
     s.info = buffer;                          /* construye una trama para transmisión */
     s.seq = next_frame_to_send;               /* inserta un número de secuencia en la
                                                  trama */
     to_physical_layer(&s);                    /* la envía a su destino */
     start_timer(s.seq);                       /* si la respuesta tarda mucho, expira el
                                                  temporizador */
     wait_for_event(&event);                   /* frame_arrival, cksum_err, timeout */
     if (event == frame_arrival){
           from_physical_layer(&s);            /* obtiene la confirmación de recepción */
           if (s.ack == next_frame_to_send){
                 stop_timer(s.ack);            /* desactiva el temporizador */
                 from_network_layer(&buffer);      /* obtiene siguiente a enviar */
                 inc(next_frame_to_send);      /* invierte next_frame_to_send */
           }
     }
  }
}

void receiver3(void)
{
  seq_nr frame_expected;
  frame r, s;
  event_type event;

  frame_expected = 0;
  while (true){
     wait_for_event(&event);                   /* posibilidades: frame_arrival, cksum_err */
     if (event == frame_arrival){              /* ha llegado una trama válida. */
          from_physical_layer(&r);             /* obtiene la trama recién llegada */
          if (r.seq == frame_expected){        /* esto es lo que hemos estado esperando. */
                 to_network_layer(&r.info);    /* pasa los datos a la capa de red */
                 inc(frame_expected);          /* para la próxima se espera el otro número
                                                  de secuencia */
          }
          s.ack = 1 – frame_expected;          /* indica la trama cuya recepción se está
                                                  confirmando */
          to_physical_layer(&s);               /* envía confirmación de recepción */
     }
  }
}
```

Figura 3-12. Protocolo de confirmación de recepción positiva con retransmisión.

Tras transmitir una trama, el emisor arranca el temporizador. Si éste ya se estaba ejecutando, se restablece para conceder otro intervalo completo de temporización. Dicho intervalo debe escogerse de modo que haya suficiente tiempo para que la trama llegue al receptor, éste la procese en el peor caso y la confirmación de recepción se regrese al emisor. Sólo cuando ha transcurrido ese intervalo de tiempo se puede suponer con seguridad que se ha perdido la trama transmitida o su confirmación de recepción, y que se debe enviar un duplicado. Si el intervalo establecido es muy pequeño, el emisor transmitirá tramas innecesarias. Si bien estas tramas adicionales no afectarán la corrección del protocolo, sí dañarán el rendimiento.

Tras transmitir una trama y arrancar el temporizador, el emisor espera que ocurra algo interesante. Hay tres posibilidades: llega una trama de confirmación de recepción sin daño, llega una trama de confirmación de recepción dañada o expira el temporizador. Si recibe una confirmación de recepción válida, el emisor obtiene el siguiente paquete de la capa de red y lo coloca en el búfer, sobrescribiendo el paquete previo. También avanza el número de secuencia. Si llega una trama dañada o no llega ninguna, ni el búfer ni el número de secuencia cambia, con el fin de que se pueda enviar un duplicado.

Cuando llega una trama válida al receptor, su número de secuencia se verifica para saber si es un duplicado. Si no lo es, se acepta, se pasa a la capa de red y se genera una confirmación de recepción. Los duplicados y las tramas dañadas no se pasan a la capa de red.

3.4 PROTOCOLOS DE VENTANA CORREDIZA

En los protocolos previos, las tramas de datos se transmiten en una sola dirección. En la mayoría de las situaciones prácticas hay necesidad de transmitir datos en ambas direcciones. Una manera de lograr una transmisión de datos dúplex total es tener dos canales de comunicación separados y utilizar cada uno para tráfico de datos símplex (en diferentes direcciones). Si se hace esto, tenemos dos circuitos físicos separados, cada uno con un canal "de ida" (para datos) y un canal "de retorno" (para confirmaciones de recepción). En ambos casos, el ancho de banda del canal usado para confirmaciones de recepción se desperdicia casi por completo. En efecto, el usuario está pagando dos circuitos, pero sólo usa la capacidad de uno.

Una mejor idea es utilizar el mismo circuito para datos en ambas direcciones. Después de todo, en los protocolos 2 y 3 ya se usaba para transmitir tramas en ambos sentidos, y el canal de retorno tiene la misma capacidad que el canal de ida. En este modelo, las tramas de datos de *A* a *B* se mezclan con las tramas de confirmación de recepción de *A* a *B*. Analizando el campo de tipo (*kind*) en el encabezado de una trama de entrada, el receptor puede saber si la trama es de datos o de confirmación de recepción.

Aunque el entrelazado de datos y de tramas de control en el mismo circuito es una mejora respecto al uso de dos circuitos físico separados, se puede lograr otra mejora. Cuando llega una trama de datos, en lugar de enviar inmediatamente una trama de control independiente, el receptor se aguanta y espera hasta que la capa de red le pasa el siguiente paquete. La confirmación de recepción se anexa a la trama de datos de salida (usando el campo *ack* del encabezado de la trama). En efecto, la confirmación de recepción viaja gratuitamente en la siguiente trama de datos de salida.

La técnica de retardar temporalmente las confirmaciones de recepción para que puedan viajar en la siguiente trama de datos de salida se conoce como **superposición** (*piggybacking*).

La ventaja principal de usar la superposición en lugar de tener tramas de confirmación de recepción independientes es un mejor aprovechamiento del ancho de banda disponible del canal. El campo *ack* del encabezado de la trama ocupa sólo unos cuantos bits, mientras que una trama aparte requeriría de un encabezado, la confirmación de recepción y una suma de verificación. Además, el envío de menos tramas implica menos interrupciones de "ha llegado trama" y tal vez menos segmentos de búfer en el receptor, dependiendo de la manera en que esté organizado el software del receptor. En el siguiente protocolo que examinaremos, el campo de superposición ocupa sólo 1 bit en el encabezado de la trama y pocas veces ocupa más de algunos bits.

Sin embargo, la superposición introduce una complicación inexistente en las confirmaciones de recepción independientes. ¿Cuánto tiempo debe esperar la capa de enlace de datos un paquete al cual superponer la confirmación de recepción? Si la capa de enlace de datos espera más tiempo del que tarda en terminar el temporizador del emisor, la trama será retransmitida, frustrando el propósito de enviar confirmaciones de recepción. Si la capa de enlace de datos fuera un oráculo y pudiera predecir el futuro, sabría cuándo se recibiría el siguiente paquete de la capa de red y podría decidir esperarlo o enviar de inmediato una confirmación de recepción independiente, dependiendo del tiempo de espera proyectado. Por supuesto, la capa de enlace de datos no puede predecir el futuro, por lo que debe recurrir a algún esquema particular para el caso, como esperar un número fijo de milisegundos. Si llega rápidamente un nuevo paquete, la confirmación de recepción se superpone a él; de otra manera, si no ha llegado ningún paquete nuevo al final de este periodo, la capa de enlace de datos manda una trama de confirmación de recepción independiente.

Los siguientes tres protocolos son bidireccionales y pertenecen a una clase llamada protocolos de **ventana corrediza.** Los tres difieren entre ellos en la eficiencia, complejidad y requerimientos de búfer, como se analizará más adelante. En ellos, al igual que en todos los protocolos de ventana corrediza, cada trama de salida contiene un número de secuencia, que va desde 0 hasta algún número máximo. Por lo general, éste es $2^n - 1$, por lo que el número de secuencia encaja perfectamente en un campo de n bits. El protocolo de ventana corrediza de parada y espera utiliza $n = 1$, y restringe los números de secuencia de 0 y 1, pero las versiones más refinadas pueden utilizar un n arbitrario.

La esencia de todos los protocolos de ventana corrediza es que, en cualquier instante, el emisor mantiene un grupo de números de secuencia que corresponde a las tramas que tiene permitido enviar. Se dice que estas tramas caen dentro de la **ventana emisora.** De manera semejante, el receptor mantiene una **ventana receptora** correspondiente al grupo de tramas que tiene permitido aceptar. La ventana del emisor y la del receptor no necesitan tener los mismos límites inferior y superior, ni siquiera el mismo tamaño. En algunos protocolos las ventanas son de tamaño fijo, pero en otros pueden crecer y disminuir a medida que se envían y reciben las tramas.

Aunque estos protocolos dan a la capa de enlace de datos mayor libertad en cuanto al orden en que puede enviar y recibir tramas, hemos conservado decididamente el requisito de que el protocolo debe entregar los paquetes a la capa de red del destino en el mismo orden en que se pasaron a la capa de enlace de datos de la máquina emisora. Tampoco hemos cambiado el requisito de que

el canal físico de comunicación es "de tipo alambre", es decir, que debe entregar todas las tramas en el orden en que fueron enviadas.

Los números de secuencia en la ventana del emisor representan tramas enviadas, o que pueden ser enviadas, pero cuya recepción aún no se ha confirmado. Cuando llega un paquete nuevo de la capa de red, se le da el siguiente número secuencial mayor, y el extremo superior de la ventana avanza en uno. Al llegar una confirmación de recepción, el extremo inferior avanza en uno. De esta manera, la ventana mantiene continuamente una lista de tramas sin confirmación de recepción. En la figura 3-13 se muestra un ejemplo.

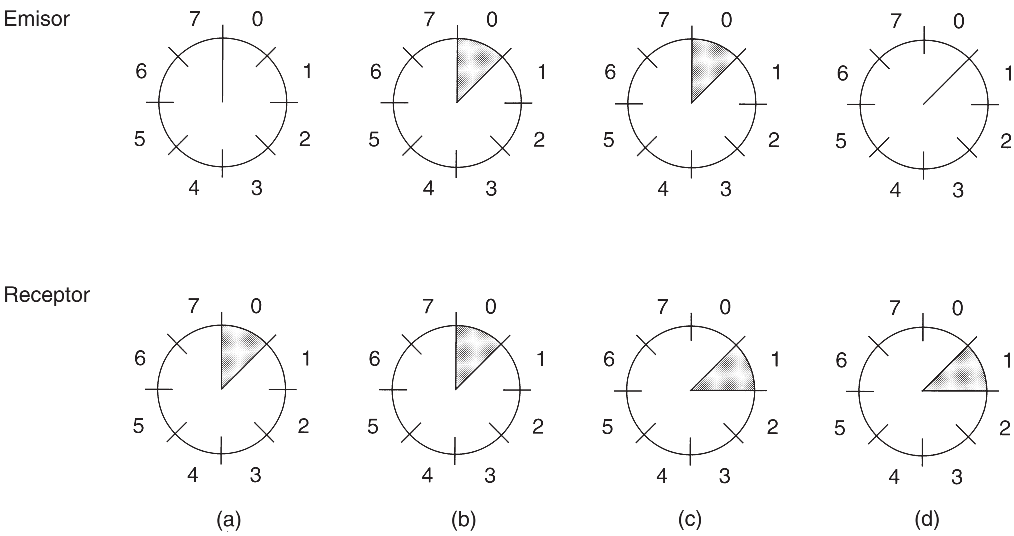


Figura 3-13. Ventana corrediza de tamaño 1, con un número de secuencia de 3 bits. (a) Al inicio. (b) Tras la transmisión de la primera trama. (c) Tras la recepción de la primera trama. (d) Tras recibir la primera confirmación de recepción.

Dado que las tramas que están en la ventana del emisor pueden perderse o dañarse en tránsito, el emisor debe mantener todas estas tramas en su memoria para su posible retransmisión. Por lo tanto, si el tamaño máximo de la ventana es *n,* el emisor necesita *n* búferes para contener las tramas sin confirmación de recepción. Si la ventana llega a crecer a su tamaño máximo, la capa de enlace de datos emisora deberá hacer que la capa de red se detenga hasta que se libere otro búfer.

La ventana de la capa de enlace de datos receptora corresponde a las tramas que puede aceptar. Toda trama que caiga fuera de la ventana se descartará sin comentarios. Cuando se recibe la trama cuyo número de secuencia es igual al extremo inferior de la ventana, se pasa a la capa de red, se genera una confirmación de recepción y se avanza la ventana en uno. A diferencia de la ventana del emisor, la ventana del receptor conserva siempre el mismo tamaño inicial. Note que

un tamaño de ventana de 1 significa que la capa de enlace de datos sólo acepta tramas en orden, pero con ventanas más grandes esto no es así. La capa de red, en contraste, siempre recibe los datos en el orden correcto, sin importar el tamaño de la ventana de la capa de enlace de datos.

En la figura 3-13 se muestra un ejemplo con un tamaño máximo de ventana de 1. Inicialmente no hay tramas pendientes, por lo que los extremos de la ventana del emisor son iguales, pero a medida que pasa el tiempo, la situación progresa como se muestra.

3.4.1 Un protocolo de ventana corrediza de un bit

Antes de lidiar con el caso general, examinemos un protocolo de ventana corrediza con un tamaño máximo de ventana de 1. Tal protocolo utiliza parada y espera, ya que el emisor envía una trama y espera su confirmación de recepción antes de transmitir la siguiente.

En la figura 3-14 se presenta tal protocolo. Como los demás, comienza por definir algunas variables. *Next_ frame_to_send* indica qué trama está tratando de enviar el emisor. De manera semejante, *frame_expected* indica qué trama espera el receptor. En ambos casos, 0 y 1 son las únicas posibilidades.

Normalmente, una de las dos capas de enlace de datos es la que comienza a transmitir la primera trama. En otras palabras, sólo uno de los programas de capa de enlace de datos debe contener las llamadas de procedimiento *to_physical_layer* y *start_timer* fuera del ciclo principal. Si ambas capas se iniciaran en forma simultánea, surgiría una situación peculiar que se analizará después. La máquina que arranca obtiene el primer paquete de su capa de red, construye una trama a partir de él y la envía. Al llegar esta (o cualquier) trama, la capa de enlace de datos receptora la revisa para saber si es un duplicado, igual que en el protocolo 3. Si la trama es la esperada, se pasa a la capa de red y la ventana del receptor se recorre hacia arriba.

El campo de confirmación de recepción contiene el número de la última trama recibida sin error. Si este número concuerda con el de secuencia de la trama que está tratando de enviar el emisor, éste sabe que ha terminado con la trama almacenada en el *búfer* y que puede obtener el siguiente paquete de su capa de red. Si el número de secuencia no concuerda, debe continuar intentando enviar la misma trama. Por cada trama que se recibe, se regresa una.

Ahora examinemos el protocolo 4 para ver qué tan flexible es ante circunstancias patológicas. Suponga que *A* está tratando de enviar su trama 0 a *B* y que *B* está tratando de enviar su trama 0 a *A*. Suponga que *A* envía una trama a *B*, pero que el intervalo de temporización de *A* es un poco corto. En consecuencia, *A* podría terminar su temporización repetidamente, enviando una serie de tramas idénticas, todas con *seq* = 0 y *ack* = 1.

Al llegar la primera trama válida a *B*, es aceptada y *frame_expected* se establece en 1. Todas las tramas subsiguientes serán rechazadas, pues *B* ahora espera tramas con el número de secuencia 1, no 0. Además, dado que los duplicados tienen *ack* = 1 y *B* aún está esperando una confirmación de recepción de 0, *B* no extraerá un nuevo paquete de su capa de red.

```
/* El protocolo 4 (de ventana corrediza) es bidireccional. */
#define MAX_SEQ 1                                  /* debe ser 1 para el protocolo 4 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"
void protocol4 (void)
{
 seq_nr next_frame_to_send;                        /* sólo 0 o 1 */
 seq_nr frame_expected;                            /* sólo 0 o 1 */
 frame r, s;                                       /* variables de trabajo */
 packet buffer;                                    /* paquete actual que se envía */
 event_type event;
 next_frame_to_send = 0;                           /* siguiente trama del flujo de
                                                      salida */
 frame_expected = 0;                               /* número de trama de llegada
                                                      esperada */
 from_network_layer(&buffer);                      /* obtiene un paquete de la capa de
                                                      red */
 s.info = buffer;                                  /* se prepara para enviar la trama
                                                      inicial */
 s.seq = next_frame_to_send;                       /* inserta el número de secuencia en
                                                      la trama */
 s.ack = 1 – frame_expected;                       /* confirmación de recepción
                                                      superpuesta */
 to_physical_layer(&s);                            /* transmite la trama */
 start_timer(s.seq);                               /* inicia el temporizador */
 while (true){
    wait_for_event(&event);                        /* frame_arrival, cksum_err o
                                                      timeout */
    if (event == frame_arrival){                   /* ha llegado una trama sin daño. */
          from_physical_layer(&r);                 /* lo obtiene */
          if(r.seq == frame_expected) {            /* maneja flujo de tramas de
                                                      entrada. */
                to_network_layer(&r.info);         /* pasa el paquete a la capa de red */
                inc(frame_expected);               /* invierte el siguiente número de
                                                      secuencia esperado */
          }
          if(r.ack == next_frame_to_send){         /* maneja flujo de tramas de
                                                      salida. */
                stop_timer(r.ack);                 /* desactiva el temporizador */
                from_network_layer(&buffer);       /* obtiene paquete nuevo de la capa
                                                      de red */
                inc(next_frame_to_send);           /* invierte el número de secuencia
                                                      del emisor */
          }
    }
    s.info = buffer;                               /* construye trama de salida */
    s.seq = next_frame_to_send;                    /* le introduce el número de
                                                      secuencia */
    s.ack = 1 – frame_expected;                    /* número de secuencia de la última
                                                      trama recibida */
    to_physical_layer(&s);                         /* transmite una trama */
    start_timer(s.seq);                            /* inicia el temporizador */
 }
}
```

Figura 3-14. Protocolo de ventana corrediza de 1 bit.

Cada vez que llega un duplicado rechazado, *B* envía a *A* una trama que contiene *seq* = 0 y *ack* = 0. Tarde o temprano una de éstas llegará correctamente a *A*, haciendo que *A* comience a enviar el siguiente paquete. Ninguna combinación de tramas perdidas o expiración de temporizadores puede hacer que el protocolo entregue paquetes duplicados a cualquiera de las capas de red, ni que omita un paquete, ni que entre en un bloqueo irreversible.

Sin embargo, si ambos lados envían de manera simultánea un paquete inicial, surge una situación peculiar. En la figura 3-15 se muestra este problema de sincronización. En la parte (a) se muestra la operación normal del protocolo. En (b) se ilustra la peculiaridad. Si *B* espera la primera trama de *A* antes de enviar la suya, la secuencia es como se muestra en (a), y todas las tramas son aceptadas. Sin embargo, si *A* y *B* inician la comunicación simultáneamente, se cruzan sus primeras tramas y las capas de enlace de datos entran en la situación (b). En (a) cada trama que llega trae un paquete nuevo para la capa de red; no hay duplicados. En (b) la mitad de las tramas contienen duplicados, aun cuando no hay errores de transmisión. Pueden ocurrir situaciones similares como resultado de la expiración prematura de temporizadores, aun cuando un lado comience evidentemente primero. De hecho, si ocurren varias expiraciones prematuras de temporizadores las tramas podrían enviarse tres o más veces.

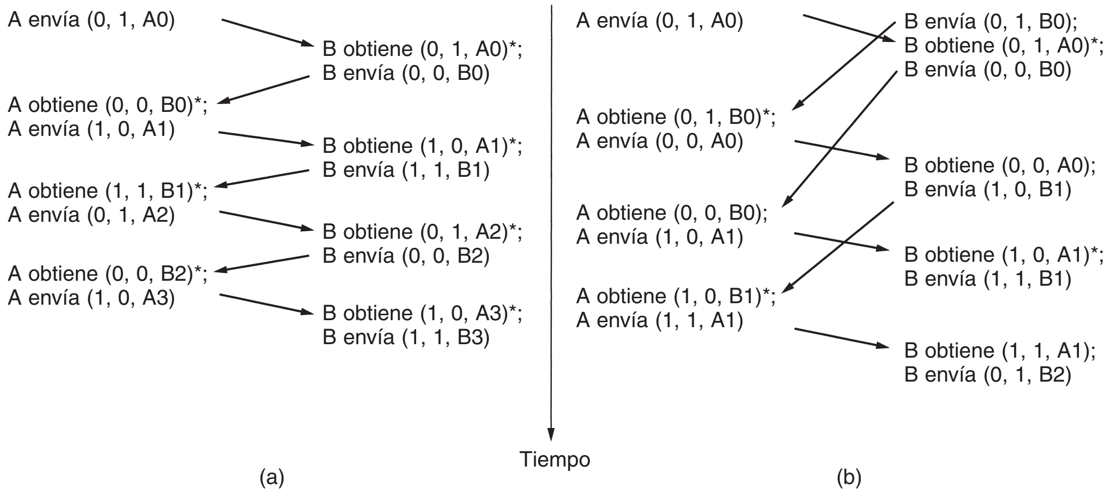


Figura 3-15. Dos escenarios para el protocolo 4. (a) Caso normal. (b) Caso anormal. La notación es (secuencia, confirmación de recepción, número de paquete). Un asterisco indica el lugar en que una capa de red acepta un paquete.

3.4.2 Protocolo que usa retroceso N

Hasta ahora hemos supuesto que el tiempo de transmisión requerido para que una trama llegue al receptor más el necesario para que la confirmación de recepción regrese es insignificante. A veces esta suposición es totalmente falsa. En estas situaciones el tiempo de viaje de ida y vuelta prolongado puede tener implicaciones importantes para la eficiencia del aprovechamiento del ancho

de banda. Por ejemplo, considere un canal de satélite de 50 kbps con un retardo de propagación de ida y vuelta de 500 mseg. Imagine que intentamos utilizar el protocolo 4 para enviar tramas de 1000 bits por medio del satélite. El emisor empieza a enviar la primera trama en $t = 0$. En $t =$ 20 mseg la trama ha sido enviada por completo. En las mejores circunstancias (sin esperas en el receptor y una trama de confirmación de recepción corta), no es sino hasta $t = 270$ mseg que la trama ha llegado por completo al receptor, y no es sino hasta $t = 520$ mseg que ha llegado la confirmación de recepción de regreso al emisor. Esto implica que el emisor estuvo bloqueado durante el 500/520 = 96% del tiempo. En otras palabras, sólo se usó el 4% del ancho de banda disponible. Queda claro que la combinación de un tiempo de tránsito grande, un ancho de banda alto y una longitud de tramas corta es desastrosa para la eficiencia.

El problema antes descrito puede verse como una consecuencia de la regla que requiere que el emisor espere una confirmación de recepción antes de enviar otra trama. Si relajamos esa restricción, se puede lograr una mejor eficiencia. Básicamente la solución está en permitir que el emisor envíe hasta w tramas antes de bloquearse, en lugar de sólo 1. Con una selección adecuada de w, el emisor podrá transmitir tramas continuamente durante un tiempo igual al tiempo de tránsito de ida y vuelta sin llenar la ventana. En el ejemplo anterior, w debe ser de cuando menos 26. El emisor comienza por enviar la trama 0, como antes. Para cuando ha terminado de enviar 26 tramas, en $t = 520$, llega la confirmación de recepción de la trama 0. A partir de entonces, las confirmaciones de recepción llegarán cada 20 mseg, por lo que el emisor siempre tendrá permiso de continuar justo cuando lo necesita. En todo momento hay 25 o 26 tramas pendientes de confirmación de recepción. Dicho de otra manera, el tamaño máximo de la ventana del emisor es de 26.

La necesidad de una ventana grande en el lado emisor se presenta cuando el producto del ancho de banda por el retardo del viaje de ida y vuelta es grande. Si el ancho de banda es alto, incluso para un retardo moderado, el emisor agotará su ventana rápidamente a menos que tenga una ventana grande. Si el retardo es grande (por ejemplo, en un canal de satélite geoestacionario), el emisor agotará su ventana incluso con un ancho de banda moderado. El producto de estos dos factores indica básicamente cuál es la capacidad del canal, y el emisor necesita la capacidad de llenarlo sin detenerse para poder funcionar con una eficiencia máxima.

Esta técnica se conoce como **canalización.** Si la capacidad del canal es de b bits/seg, el tamaño de la trama de l bits y el tiempo de propagación de ida y vuelta de R segundos, el tiempo requerido para transmitir una sola trama es de l/b segundos. Una vez que ha sido enviado el último bit de una trama de datos, hay un retardo de $R/2$ antes de que llegue ese bit al receptor y un retardo de cuando menos $R/2$ para que la confirmación de recepción llegue de regreso, lo que da un retardo total de R. En parada y espera, la línea está ocupada durante l/b e inactiva durante R, dando

$$\text{una utilización de la línea de} = l/(l + bR)$$

Si $l < bR$, la eficiencia será menor que 50%. Ya que siempre hay un retardo diferente de cero para que la confirmación de recepción se propague de regreso, en principio la canalización puede servir para mantener ocupada la línea durante este intervalo, pero si el intervalo es pequeño, la complejidad adicional no justifica el esfuerzo.

El envío de tramas en canalización por un canal de comunicación inestable presenta problemas serios. Primero, ¿qué ocurre si una trama a la mitad de una serie larga se daña o pierde? Llegarán grandes cantidades de tramas sucesivas al receptor antes de que el emisor se entere de que algo anda mal. Cuando llega una trama dañada al receptor, obviamente debe descartarse, pero, ¿qué debe hacerse con las tramas correctas que le siguen? Recuerde que la capa de enlace de datos receptora está obligada a entregar paquetes a la capa de red en secuencia. En la figura 3-16 se muestran los efectos de la canalización en la recuperación de un error. A continuación lo analizaremos en detalle.

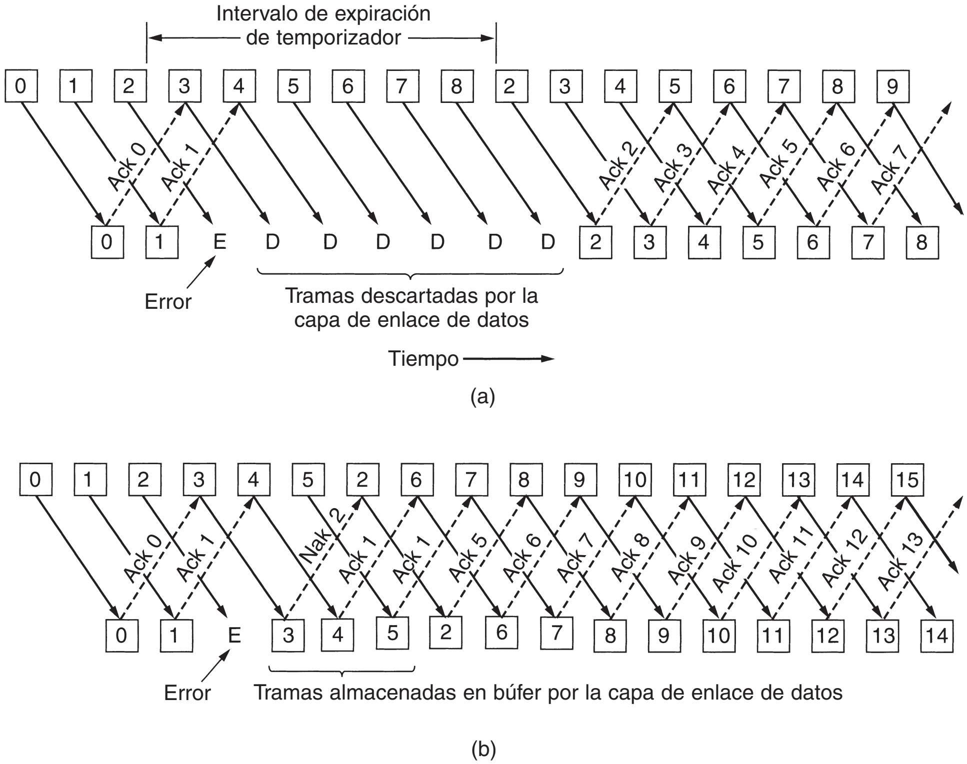


Figura 3-16. Canalización y recuperación de un error. (a) Efecto de un error cuando el tamaño de la ventana del receptor es de 1. (b) Efecto de un error cuando el tamaño de la ventana del receptor es grande.

Hay dos métodos básicos para manejar los errores durante la canalización. Una manera, llamada **retroceso n,** es que el receptor simplemente descarte todas las tramas subsecuentes, sin enviar confirmaciones de recepción para las tramas descartadas. Esta estrategia corresponde a una ventana de recepción de tamaño 1. En otras palabras, la capa de enlace de datos se niega a aceptar cualquier trama excepto la siguiente que debe entregar a la capa de red. Si la ventana del emisor se llena antes de terminar el temporizador, el canal comenzará a vaciarse. En algún momento, el emisor terminará de esperar y retransmitirá en orden todas las tramas cuya recepción aún no se

haya confirmado, comenzando por la dañada o perdida. Esta estrategia puede desperdiciar bastante ancho de banda si la tasa de errores es alta.

En la figura 3-16(a) se muestra el retroceso n en el caso en que la ventana del receptor es grande. Las tramas 0 y 1 se reciben y confirman de manera correcta. Sin embargo, la trama 2 se daña o pierde. El emisor, no consciente de este problema, continúa enviando tramas hasta que expira el temporizador para la trama 2. Después retrocede a la trama 2 y comienza con ella, enviando 2, 3, 4, etcétera, nuevamente.

La otra estrategia general para el manejo de errores cuando las tramas se colocan en canalizaciones se conoce como **repetición selectiva.** Cuando se utiliza, se descarta una trama dañada recibida, pero las tramas en buen estado recibidas después de ésa se almacenan en el búfer. Cuando el emisor termina, sólo la última trama sin confirmación se retransmite. Si la trama llega correctamente, el receptor puede entregar a la capa de red, en secuencia, todas las tramas que ha almacenado en el búfer. La repetición selectiva con frecuencia se combina con el hecho de que el receptor envíe una confirmación de recepción negativa (NAK) cuando detecta un error, por ejemplo, cuando recibe un error de suma de verificación o una trama en desorden. Las confirmaciones de recepción negativas estimulan la retransmisión antes de que el temporizador correspondiente expire y, por lo tanto, mejoran el rendimiento.

En la figura 3-16(b), las tramas 0 y 1 se vuelven a recibir y confirmar correctamente y la trama 2 se pierde. Cuando la trama 3 llega al receptor, su capa de enlace de datos observa que falta una trama, por lo que regresa una NAK para la trama 2 pero almacena la trama 3. Cuando las tramas 4 y 5 llegan, también son almacenadas por la capa de enlace de datos en lugar de pasarse a la capa de red. En algún momento, la NAK 2 llega al emisor, que inmediatamente reenvía la trama 2. Cuando llega, la capa de enlace de datos ahora tiene 2, 3, 4 y 5 y ya las puede pasar a la capa de red en el orden correcto. También puede confirmar la recepción de todas las tramas hasta, e incluyendo, la 5, como se muestra en la figura. Si la NAK se perdiera, en algún momento el temporizador del emisor expirará para la trama 2 y la enviará (sólo a ella), pero eso puede tardar un poco más. En efecto, la NAK acelera la retransmisión de una trama específica.

La repetición selectiva corresponde a una ventana del receptor mayor que 1. Cualquier trama dentro de la ventana puede ser aceptada y mantenida en el búfer hasta que todas las que le preceden hayan sido pasadas a la capa de red. Esta estrategia puede requerir cantidades grandes de memoria en la capa de enlace de datos si la ventana es grande.

Estas dos estrategias alternativas son intercambios entre el ancho de banda y el espacio de búfer en la capa de enlace de datos. Dependiendo de qué recurso sea más valioso, se puede utilizar uno o el otro. En la figura 3-17 se muestra un protocolo de canalización en el que la capa de enlace de datos receptora sólo acepta tramas en orden; las tramas siguientes a un error son descartadas. En este protocolo hemos omitido por primera vez el supuesto de que la capa de red siempre tiene un suministro infinito de paquetes que enviar. Cuando la capa de red tiene un paquete para enviar, puede causar la ocurrencia de un evento *network_layer_ready*. Sin embargo, para poder cumplir la regla de control de flujo de que no debe haber más de *MAX_SEQ* tramas sin confirmación de recepción pendientes en cualquier momento, la capa de enlace de datos debe poder prohibir a la capa de red que la moleste con más trabajo. Los procedimientos de biblioteca *enable_network_layer* y *disable_network_layer* llevan a cabo esta función.

```
/* El protocolo 5 (retroceso n) permite múltiples tramas pendientes. El emisor podría
   enviar hasta MAX_SEQ tramas sin esperar una confirmación de recepción. Además, a
   diferencia de los primeros protocolos, no se supone que la capa de red debe tener
   siempre un paquete nuevo. En vez de ello, la capa de red activa un evento
   network_layer_ready cuando hay un paquete por enviar. */

#define MAX_SEQ 7                       /* debe ser 2^n - 1 */
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready} event_type;
#include "protocol.h"

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
/* Devuelve true si a <=b < c de manera circular, y false en caso contrario. */
  if (((a <= b) && (b < c)) ¦¦ ((c < a) && (a <= b)) ¦¦ ((b < c) && (c < a)))
        return(true);
     else
        return(false);
}

static void send_data(seq_nr frame_nr, seq_nr frame_expected, packet buffer[])
{
/* Elabora y envía una trama de datos. */
  frame s;                              /* variable de trabajo */

  s.info = buffer[frame_nr];            /* inserta el paquete en la trama */
  s.seq = frame_nr;                     /* inserta un número de secuencia en la trama */
  s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);      /* ack superpuesta*/
  to_physical_layer(&s);                /* transmite la trama */
  start_timer(frame_nr);                /* inicia la ejecución del temporizador */
}

void protocol5(void)
{
  seq_nr next_frame_to_send;            /* MAX_SEQ > 1; utilizado para flujo de salida */
  seq_nr ack_expected;                  /* la trama más vieja hasta el momento no
                                           confirmada */
  seq_nr frame_expected;                /* siguiente trama esperada en el flujo de
                                           entrada */
  frame r;                              /* variable de trabajo */
  packet buffer[MAX_SEQ + 1];           /* búferes para el flujo de salida */
  seq_nr nbuffered;                     /* # de búferes de salida actualmente en uso */
  seq_nr i;                             /* utilizado para indexar en el arreglo de
                                           búferes */
  event_type event;

  enable_network_layer();               /* permite eventos network_layer_ready */
  ack_expected = 0;                     /* siguiente ack esperado en el flujo
                                           de entrada */
  next_frame_to_send = 0;               /* siguiente trama de salida */
  frame_expected = 0;                   /* número de trama de entrada esperada */
  nbuffered = 0;                        /* en principio no hay paquetes en búfer */
  while (true) {
   wait_for_event(&event);              /* cuatro posibilidades: vea event_type al
                                           principio */
```

```
   switch(event) {
    case network_layer_ready:        /* la capa de red tiene un paquete para enviar */
        /* Acepta, guarda y transmite una trama nueva. */
        from_network_layer(&buffer[next_frame_to_send]); /* obtiene un paquete nuevo */
        nbuffered = nbuffered + 1;   /* expande la ventana del emisor */
        send_data(next_frame_to_send, frame_expected, buffer); /* transmite la trama */
        inc(next_frame_to_send);     /* avanza el límite superior de la ventana del
                                        emisor */
        break;

    case frame_arrival:              /* ha llegado una trama de datos o de control */
        from_physical_layer(&r);     /* obtiene una trama entrante de la capa física */

        if (r.seq == frame_expected) {
              /* Las tramas se aceptan sólo en orden. */
              to_network_layer(&r.info); /* pasa el paquete a la capa de red */
              inc(frame_expected);   /* avanza el límite inferior de la ventana del
                                        receptor */
        }

        /* Ack n implica n - 1, n - 2, etc.  Verificar esto. */
        while (between(ack_expected, r.ack, next_frame_to_send)) {
              /* Maneja la ack superpuesta. */
              nbuffered = nbuffered - 1; /* una trama menos en el búfer */
              stop_timer(ack_expected);  /* la trama llegó intacta; detener el
                                            temporizador */
              inc(ack_expected);     /* reduce la ventana del emisor */
        }
        break;

    case cksum_err: break;           /* ignora las tramas erróneas */

    case timeout:                    /* problemas; retransmite todas las tramas
                                        pendientes */
        next_frame_to_send = ack_expected; /* inicia aquí la retransmisión */
        for (i = 1; i <= nbuffered; i++) {
               send_data(next_frame_to_send, frame_expected, buffer);/* reenvía la
                                                                        trama 1 */
               inc(next_frame_to_send);   /* se prepara para enviar la siguiente */
        }

  }

  if (nbuffered < MAX_SEQ)
        enable_network_layer();
  else
        disable_network_layer();
 }
}
```

Figura 3-17. Protocolo de ventana corrediza con retroceso n.

Note que pueden como máximo estar pendientes *MAX_SEQ* tramas, y no *MAX_SEQ* + 1 en cualquier momento, aun cuando haya *MAX_SEQ* + 1 números de secuencia diferentes: 0, 1, 2,..., *MAX_SEQ*. Para ver por qué es necesaria esta restricción, considere la siguiente situación con *MAX_SEQ* = 7.

1. El emisor envía las tramas 0 a 7.
2. En algún momento llega al emisor una confirmación de recepción, superpuesta, para la trama 7.
3. El emisor envía otras ocho tramas, nuevamente con los números de secuencia 0 a 7.
4. Ahora llega otra confirmación de recepción, superpuesta, para la trama 7.

La pregunta es: ¿llegaron con éxito las ocho tramas que correspondían al segundo bloque o se perdieron (contando como pérdidas los rechazos siguientes a un error)? En ambos casos el receptor podría estar enviando la trama 7 como confirmación de recepción. El emisor no tiene manera de saberlo. Por esta razón, el número máximo de tramas pendientes debe restringirse a *MAX_SEQ*.

Aunque el protocolo 5 no pone en el búfer las tramas que llegan tras un error, no escapa del problema de los búferes por completo. Dado que un emisor puede tener que retransmitir en un momento futuro todas las tramas no confirmadas, debe retener todas las tramas retransmitidas hasta saber con certeza que han sido aceptadas por el receptor. Al llegar una confirmación de recepción para la trama n, las tramas $n - 1$, $n - 2$, y demás, se confirman de manera automática. Esta propiedad es especialmente importante cuando algunas tramas previas portadoras de confirmaciones de recepción se perdieron o dañaron. Cuando llega una confirmación de recepción, la capa de enlace de datos revisa si se pueden liberar búferes. Si esto es posible (es decir, hay espacio disponible en la ventana), ya puede permitirse que una capa de red previamente bloqueada produzca más eventos *network_layer_ready*.

Para este protocolo damos por hecho que siempre hay tráfico de regreso en el que se pueden superponer confirmaciones de recepción. Si no lo hay, no es posible enviar confirmaciones de recepción. El protocolo 4 no necesita este supuesto debido a que envía una trama cada vez que recibe una trama, incluso si ya ha enviado la trama. En el siguiente protocolo resolveremos el problema del tráfico de una vía de una forma elegante.

Debido a que este protocolo tiene múltiples tramas pendientes, necesita lógicamente múltiples temporizadores, uno por cada trama pendiente. Cada trama termina de temporizar independientemente de todas las demás. Todos estos temporizadores pueden simularse fácilmente en software, usando un solo reloj de hardware que produzca interrupciones periódicas. Las terminaciones de temporización pendientes forman una lista enlazada, en la que cada nodo de la lista indica la cantidad de pulsos de reloj que faltan para que expire el temporizador, el número de la trama temporizada y un apuntador al siguiente nodo.

Como ilustración de una manera de implementar los temporizadores, considere el ejemplo de la figura 3-18(a). Suponga que el reloj pulsa una vez cada 100 mseg. Inicialmente la hora real es 10:00:00.0 y hay tres terminaciones pendientes, a las 10:00:00.5, 10:00:01.3 y 10:00:01.9.

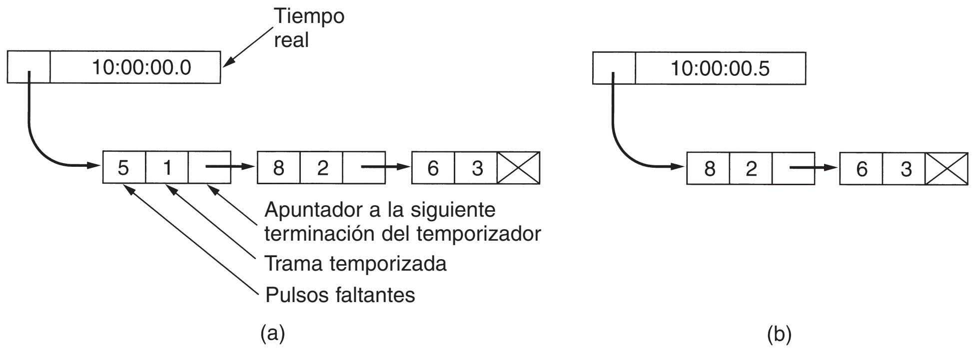


Figura 3-18. Simulación de múltiples temporizadores en software.

Cada vez que pulsa el reloj de hardware, se actualiza el tiempo real y el contador de pulsos a la cabeza de la lista se decrementa. Al llegar a cero el contador de pulsos, se causa una terminación y se retira el nodo de la lista, como se muestra en la figura 3-18(b). Aunque esta organización requiere que la lista se examine al llamar a *start_timer* o a *stop_timer*, no requiere mucho trabajo por pulso. En el protocolo 5 estas dos rutinas tienen un parámetro que indica la trama a temporizar.

3.4.3 Protocolo que utiliza repetición selectiva

El protocolo 5 funciona bien si los errores son poco frecuentes, pero si la línea es mala, se desperdicia mucho ancho de banda en las tramas retransmitidas. Una estrategia alterna para el manejo de errores es permitir que el receptor acepte y coloque en búferes las tramas que siguen a una trama dañada o perdida. Tal protocolo no rechaza tramas simplemente porque se dañó o se perdió una trama anterior.

En este protocolo, tanto el emisor como el receptor mantienen una ventana de números de secuencia aceptables. El tamaño de la ventana del emisor comienza en 0 y crece hasta un máximo predefinido, *MAX_SEQ*. La ventana del receptor, en cambio, siempre es de tamaño fijo e igual a *MAX_SEQ*. El receptor tiene un búfer reservado para cada número de secuencia en su ventana fija. Cada búfer tiene un bit asociado (*arrived*) que indica si el búfer está lleno o vacío. Cuando llega una trama, su número de secuencia es revisado por la función *between* para ver si cae dentro de la ventana. De ser así, y no ha sido recibida aún, se acepta y almacena. Esta acción se lleva a cabo sin importar si la trama contiene el siguiente paquete esperado por la capa de red. Por supuesto, debe mantenerse dentro de la capa de enlace de datos sin entregarse a la capa de red hasta que todas las tramas de número menor hayan sido entregadas a la capa de red en el orden correcto. En la figura 3-19 se presenta un protocolo que usa este algoritmo.

```
/* El protocolo 6 (repetición selectiva) acepta tramas en desorden y pasa paquetes en
   orden a la capa de red. Cada trama pendiente tiene un temporizador asociado. Cuando
   el temporizador expira, a diferencia de lo que ocurre en el protocolo 5, sólo se
   retransmite esa trama, no todas las que están pendientes. */

#define MAX_SEQ 7                   /* debe ser 2^n – 1 */
#define NR_BUFS ((MAX_SEQ + 1)/2)
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready, ack_timeout}
  event_type;
#include "protocol.h"
boolean no_nak = true;              /* aún no se ha enviado un nak */
seq_nr oldest_frame = MAX_SEQ + 1; /* el valor inicial es sólo para el simulador */

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
/* Parecido a lo que ocurre en el protocolo 5, pero más corto y confuso. */
  return ((a <= b) && (b < c)) ¦¦ ((c < a) && (a <= b)) ¦¦ ((b < c) && (c < a));
}

static void send_frame(frame_kind fk, seq_nr frame_nr, seq_nr frame_expected, packet
  buffer[])
{
/* Construye y envía una trama de datos, ack o nak. */
  frame s;                          /* variable de trabajo */

  s.kind = fk;                      /* kind == datos, ack o nak */
  if (fk == data) s.info = buffer[frame_nr % NR_BUFS];
  s.seq = frame_nr;                 /* sólo tiene importancia para las tramas de datos */
  s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);
  if (fk == nak) no_nak = false;    /* un nak por trama, por favor */
  to_physical_layer(&s);            /* transmite la trama */
  if (fk == data) start_timer(frame_nr % NR_BUFS);
  stop_ack_timer();                 /* no es necesario para tramas ack separadas */
}

void protocol6(void)
{
  seq_nr ack_expected;              /* límite inferior de la ventana del emisor */
  seq_nr next_frame_to_send;        /* límite superior de la ventana del emisor + 1 */
  seq_nr frame_expected;            /* límite inferior de la ventana del receptor */
  seq_nr too_far;                   /* límite superior de la ventana del receptor + 1 */
  int i;                            /* índice en el grupo de búferes */
  frame r;                          /* variable de trabajo*/
  packet out_buf[NR_BUFS];          /* búferes para el flujo de salida */
  packet in_buf[NR_BUFS];           /* búferes para el flujo de entrada */
  boolean arrived[NR_BUFS];         /* mapa de bits de entrada */
  seq_nr nbuffered;                 /* cuántos búferes de salida se utilizan
                                       actualmente */
  event_type event;

  enable_network_layer();           /* inicializar */
  ack_expected = 0;                 /* siguiente ack esperada en el flujo de entrada */
  next_frame_to_send = 0;           /* número de la siguiente trama de salida */
  frame_expected = 0;
  too_far = NR_BUFS;
  nbuffered = 0;                    /* inicialmente no hay paquetes en el búfer */
  for (i = 0; i < NR_BUFS; i++) arrived[i] = false;

  while (true) {
   wait_for_event(&event);          /* cinco posibilidades: vea event_type al principio */
   switch(event) {
    case network_layer_ready: /* acepta, guarda y transmite una trama nueva */
        nbuffered = nbuffered + 1;  /* expande la ventana */
```

```
        from_network_layer(&out_buf[next_frame_to_send % NR_BUFS]);/* obtiene un
                                                                   paquete nuevo */
        send_frame(data, next_frame_to_send, frame_expected, out_buf);/* transmite la
                                                                     trama */
        inc(next_frame_to_send); /* avanza el límite superior de la ventana */
        break;

    case frame_arrival:           /* ha llegado una trama de datos o de control */
        from_physical_layer(&r);/* obtiene una trama entrante de la capa física */
        if (r.kind == data) {
              /* Ha llegado una trama no dañada. */
              if ((r.seq != frame_expected) && no_nak)
                 send_frame(nak, 0, frame_expected, out_buf); else start_ack_timer();
              if (between(frame_expected, r.seq, too_far) && (arrived[r.seq%NR_BUFS]
                   == false)) {
                   /* Las tramas se podrían aceptar en cualquier orden. */
                   arrived[r.seq % NR_BUFS] = true;     /* marca como lleno el búfer */
                   in_buf[r.seq % NR_BUFS] = r.info;    /* inserta datos en el búfer */
                   while (arrived[frame_expected % NR_BUFS]) {
                        /* Pasa tramas y avanza la ventana. */
                        to_network_layer(&in_buf[frame_expected % NR_BUFS]);
                        no_nak = true;
                        arrived[frame_expected % NR_BUFS] = false;
                        inc(frame_expected);      /* avanza el límite inferior de la
                                                          ventana del receptor */
                        inc(too_far);      /* avanza el límite superior de la
                                              ventana del receptor */
                        start_ack_timer(); /* para saber si es necesaria una ack
                                              separada */
                   }
              }
        }
        if((r.kind==nak) && between(ack_expected,(r.ack+1)%(MAX_SEQ+1),
           next_frame_to_send))
              send_frame(data, (r.ack+1) % (MAX_SEQ + 1), frame_expected, out_buf);

        while (between(ack_expected, r.ack, next_frame_to_send)) {
              nbuffered = nbuffered - 1;   /* maneja la ack superpuesta */
              stop_timer(ack_expected % NR_BUFS);  /* la trama llega intacta */
              inc(ack_expected);  /* avanza el límite inferior de la ventana del emisor */
        }
        break;

    case cksum_err:
        if (no_nak) send_frame(nak, 0, frame_expected, out_buf); /* trama dañada */
        break;

    case timeout:
        send_frame(data, oldest_frame, frame_expected, out_buf);  /* hacemos que expire
                                                                el temporizador */
        break;

    case ack_timeout:
        send_frame(ack,0,frame_expected, out_buf);  /* expira el temporizador de ack;
                                                       envía ack */
  }
  if (nbuffered < NR_BUFS) enable_network_layer(); else disable_network_layer();
 }
}
```

Figura 3-19. Protocolo de ventana corrediza con repetición selectiva.

La recepción no secuencial introduce ciertos problemas que no se presentan en los protocolos en los que las tramas sólo se aceptan en orden. Podemos ilustrar el problema fácilmente con un ejemplo. Suponga que tenemos un número de secuencia de tres bits, por lo que se permite al emisor enviar hasta siete tramas antes de que se le exija que espere una confirmación de recepción. Inicialmente las ventanas del emisor y del receptor están como se muestra en la figura 3-20(a). El emisor ahora transmite las tramas 0 a 6. La ventana del receptor le permite aceptar cualquier trama con un número de secuencia entre 0 y 6, inclusive. Las siete tramas llegan correctamente, por lo que el receptor confirma su recepción y avanza su ventana para permitir la recepción de 7, 0, 1, 2, 3, 4 o 5, como se muestra en la figura 3-20(b). Los siete búferes se marcan como vacíos.

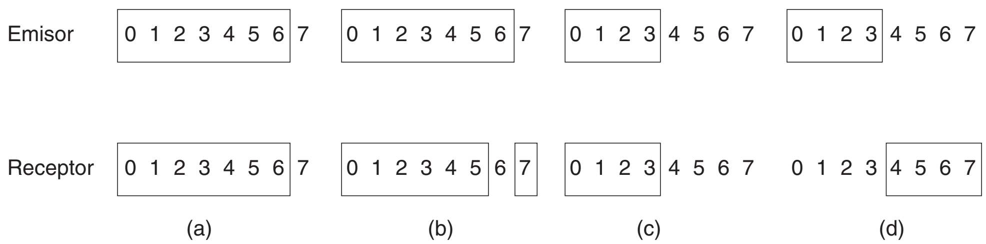


Figura 3-20. (a) Situación original con una ventana de tamaño 7. (b) Después de que se han enviado y recibido siete tramas, pero su recepción no se ha confirmado. (c) Situación inicial con un tamaño de ventana de 4. (d) Después de que se han enviado y recibido cuatro tramas, pero su recepción no se ha confirmado.

Es en este punto en el que aparece un desastre en la forma de un rayo que cae en el poste telefónico, borrando todas las confirmaciones de recepción. En algún momento termina el temporizador del emisor y retransmite la trama 0. Cuando esta trama llega al receptor, se efectúa una verificación para saber si está dentro de la ventana del receptor. Desgraciadamente, en la figura 3-20(b), la trama 0 está dentro de la nueva ventana, por lo que se acepta. El receptor envía una confirmación de recepción, superpuesta, para la trama 6, ya que se han recibido de la 0 a la 6.

El emisor se entera con beneplácito que todas sus tramas transmitidas han llegado de manera correcta, por lo que avanza su ventana y envía de inmediato las tramas 7, 0, 1, 2, 3, 4 y 5. El receptor aceptará la trama 7 y el paquete de ésta se pasará directamente a la capa de red. Inmediatamente después, la capa de enlace de datos receptora revisa si ya tiene una trama 0 válida, descubre que sí y pasa el paquete que contiene a la capa de red. En consecuencia, la capa de red obtiene un paquete incorrecto, y falla el protocolo.

La esencia del problema es que una vez que el receptor ha avanzado su ventana, el nuevo intervalo de números de secuencia válidos se traslapa con el anterior. En consecuencia, el siguiente grupo de tramas podría ser de tramas duplicadas (si se perdieron todas las confirmaciones de recepción) o de nuevas (si se recibieron todas las confirmaciones de recepción). El pobre receptor no tiene manera de distinguir entre estos dos casos.

La salida de este dilema es asegurarse que, una vez que el emisor haya avanzado su ventana, no haya traslape con la ventana original. Para asegurarse de que no haya traslape, el tamaño máximo de la ventana debe ser cuando menos de la mitad del intervalo de los números de secuencia, como en las figuras 3-20(c) y 3-20(d). Por ejemplo, si se utilizan 4 bits para los números de secuencia, éstos tendrán un intervalo de 0 a 15. Sólo ocho tramas sin confirmación de recepción deben estar pendientes en cualquier instante. De esa manera, si el receptor apenas ha aceptado las tramas 0 a 7 y ha avanzado su ventana para permitir la aceptación de las tramas 8 a 15, puede distinguir sin ambigüedades si las tramas subsiguientes son retransmisiones (0 a 7) o nuevas (8 a 15). En general, el tamaño de la ventana para el protocolo 6 será (*MAX_SEQ* + 1)/2. Por lo tanto, para números de secuencia de 3 bits, el tamaño de ventana es 4.

Una pregunta interesante es: ¿cuántos búferes deberá tener el receptor? En ninguna circunstancia puede aceptar tramas cuyos números de secuencia estén por debajo del extremo inferior o por encima del extremo superior de la ventana. En consecuencia, el número de búferes necesarios es igual al tamaño de la ventana, no al intervalo de números de secuencia. En el ejemplo anterior de un número de secuencia de 4 bits, se requieren ocho búferes, numerados del 0 al 7. Cuando llega la trama i, se coloca en el búfer i mod 8. Observe que, aun cuando i e $(i + 8)$ mod 8 están "compitiendo" por el mismo búfer, nunca están dentro de la ventana al mismo tiempo, pues ello implicaría un tamaño de ventana de cuando menos 9.

Por la misma razón, el número de temporizadores requeridos es igual al número de búferes, no al tamaño del espacio de secuencia. Efectivamente, hay un temporizador asociado a cada búfer. Cuando termina el temporizador, el contenido del búfer se retransmite.

En el protocolo 5 se supone de manera implícita que el canal está fuertemente cargado. Cuando llega una trama, no se envía de inmediato la confirmación de recepción. En cambio, esta última se superpone en la siguiente trama de datos de salida. Si el tráfico de regreso es ligero, la confirmación de recepción se detendrá durante un periodo largo. Si hay mucho tráfico en una dirección y no hay tráfico en la otra, sólo se envían *MAX_SEQ* paquetes y luego se bloquea el protocolo, que es por lo cual dimos por hecho que siempre había tráfico de regreso.

En el protocolo 6 se corrige este problema. Tras llegar una trama de datos en secuencia, se arranca un temporizador auxiliar mediante *start_ack_timer*. Si no se ha presentado tráfico de regreso antes de que termine este temporizador, se envía una trama de confirmación de recepción independiente. Una interrupción debida al temporizador auxiliar es conoce como evento *ack_timeout*. Con este arreglo, ahora es posible el flujo de tráfico unidireccional, pues la falta de tramas de datos de regreso a las que puedan superponerse las confirmaciones de recepción ya no es un obstáculo. Sólo existe un temporizador auxiliar, y si *start_ack_timer* se invoca mientras el temporizador se está ejecutando, se restablece a un intervalo completo de temporización de la confirmación de recepción.

Es indispensable que el tiempo asociado al temporizador auxiliar sea notablemente más corto que el del temporizador usado para la terminación de tramas de datos. Esta condición es necesaria para asegurarse de que la confirmación de recepción de una trama correctamente recibida llegue antes de que el emisor termine su temporización y retransmita la trama.

El protocolo 6 utiliza una estrategia más eficiente que el 5 para manejar los errores. Cuando el receptor tiene razones para suponer que ha ocurrido un error, envía al emisor una trama de confirmación de recepción negativa (NAK). Tal trama es una solicitud de retransmisión de la trama especificada en la NAK. Hay dos casos en los que el receptor debe sospechar: cuando llega una trama dañada, o cuando llega una trama diferente de la esperada (pérdida potencial de la trama). Para evitar hacer múltiples solicitudes de retransmisión de la misma trama perdida, el receptor debe saber si ya se ha enviado una NAK para una trama dada. La variable *no_nak* del protocolo 6 es true si no se ha enviado todavía ninguna NAK para *frame_expected*. Si la NAK se altera o se pierde no hay un daño real, pues de todos modos el emisor terminará su temporizador en algún momento y retransmitirá la trama perdida. Si llega la trama equivocada después de haberse enviado y de perderse una NAK, *no_nak* será true y el temporizador auxiliar arrancará. Cuando termine, se enviará una ACK para resincronizar el estado actual del emisor con el del receptor.

En algunas situaciones, el tiempo requerido para que una trama se propague a su destino, sea procesada ahí y regrese la confirmación de recepción es (casi) constante. En estos casos, el emisor puede ajustar su temporizador para que apenas sea mayor que el intervalo de tiempo esperado entre el envío de una trama y la recepción de su confirmación. Sin embargo, si este tiempo es muy variable, el emisor se enfrenta a la decisión de establecer el intervalo en un valor pequeño (y arriesgarse a retransmisiones innecesarias) o de establecerlo en un valor grande (quedándose inactivo por un periodo largo tras un error).

Ambas opciones desperdician ancho de banda. Si el tráfico de regreso es esporádico, el tiempo antes de la confirmación de recepción será irregular, siendo más corto al haber tráfico de regreso y mayor al no haberlo. El tiempo de procesamiento variable en el receptor también puede ser un problema aquí. En general, cuando la desviación estándar del intervalo de confirmación de recepción es pequeña en comparación con el intervalo mismo, el temporizador puede hacerse "justo" y las NAKs no serán útiles. De otra manera, el temporizador deberá hacerse "holgado", y las NAKs pueden acelerar apreciablemente la retransmisión de tramas perdidas o dañadas.

Un aspecto muy relacionado con el asunto de la terminación de los temporizadores y las NAKs es cómo determinar qué trama causó una terminación del temporizador. En el protocolo 5 siempre es *ack_expected*, pues es la más antigua. En el protocolo 6 no hay ninguna manera sencilla de determinar quién ha terminado el temporizador. Suponga que ya se transmitieron las tramas 0 a 4, de modo que la lista de tramas pendientes es 01234, en orden de la más antigua a la más nueva. Ahora imagine que 0 termina su temporizador, se transmite 5 (una trama nueva), 1 termina su temporizador, 2 termina su temporizador y se transmite 6 (otra trama nueva). En este punto, la lista de tramas pendientes es 3405126, de la más antigua a la más nueva. Si todo el tráfico de entrada se pierde durante un rato (es decir, las tramas que llevan las confirmaciones de recepción), las siete tramas pendientes terminarán su temporizador en ese orden.

Para evitar que el ejemplo se complique aún más, no hemos mostrado la administración de los temporizadores. En cambio, suponemos que la variable *oldest_frame* se establece en el momento de terminación del temporizador para indicar la trama cuyo temporizador ha terminado.

3.5 VERIFICACIÓN DE LOS PROTOCOLOS

Los protocolos que se utilizan en la práctica, y los programas que los implementan, con frecuencia son complicados. En consecuencia, se requiere mucha investigación para encontrar técnicas matemáticas formales con las cuales especificar y verificar los protocolos. En las siguientes secciones veremos algunos modelos y técnicas. Aunque estamos estudiándolos en el contexto de la capa de enlace de datos, también son aplicables a otras capas.

3.5.1 Modelos de máquinas de estado finito

Un concepto clave empleado en muchos modelos de protocolos es el de la **máquina de estados finitos.** Con esta técnica, cada **máquina de protocolo** (es decir, emisor o receptor) siempre está en un estado específico en cualquier instante. Su estado consiste en todos los valores de sus variables, incluido el contador de programa.

En la mayoría de los casos puede agruparse un gran número de estados a fin de analizarlos. Por ejemplo, considerando el receptor del protocolo 3, podemos abstraer dos estados importantes de todos los posibles: en espera de la trama 0 y en espera de la trama 1. Todos los demás estados pueden considerarse como transitorios: pasos en el camino hacia uno de los estados principales. Por lo general, los estados se escogen para que sean aquellos instantes en que la máquina de protocolo está esperando que ocurra el siguiente evento [es decir, la ejecución de la llamada de procedimiento *wait*(*event*) en nuestros ejemplos]. En este punto, el estado de la máquina de protocolo está determinado por completo por los estados de sus variables. El número de estados es entonces 2^n, donde n es el número de bits necesarios para representar todas las variables combinadas.

El estado del sistema completo es la combinación de todos los estados de las dos máquinas de protocolos y del canal. El estado del canal está determinado por su contenido. Usando nuevamente el protocolo 3 como ejemplo, el canal tiene cuatro posibles estados: una trama cero o una trama 1 viajando del emisor al receptor, una trama de confirmación de recepción que va en el otro sentido o un canal vacío. Si modelamos el emisor y el receptor como si ambos tuvieran dos estados, todo el sistema tiene 16 estados diferentes.

Vale la pena decir algo sobre el estado del canal. El concepto de una trama que está “en el canal” es, por supuesto, una abstracción. Lo que queremos decir en realidad es que es posible que una trama se haya recibido, pero no procesado, en el destino. Una trama permanece “en el canal” hasta que la máquina de protocolo ejecuta *FromPhysicalLayer* y la procesa.

De cada estado hay cero o más **transiciones** posibles a otros estados. Las transiciones ocurren cuando sucede algún evento. Para una máquina de protocolo, podría ocurrir una transición al enviar una trama, al llegar una trama, al terminar un temporizador, al ocurrir una interrupción, etcétera. Para el canal, los eventos típicos son la inserción de una trama nueva en el canal por una máquina de protocolo, la entrega de una trama a una máquina de protocolo o la pérdida de una trama debido a una ráfaga de ruido. Dada una descripción completa de las máquinas de protocolo y las características del canal, es posible dibujar un grafo dirigido que muestre todos los estados como nodos y las transiciones como arcos dirigidos.

Un estado en particular se designa como **estado inicial.** Este estado corresponde a la descripción del sistema cuando comienza a funcionar, o en algún punto conveniente poco después. Desde el estado inicial pueden alcanzarse algunos, o quizá todos, los demás estados mediante una secuencia de transiciones. Usando técnicas bien conocidas de la teoría de grafos (por ejemplo, calculando el cierre transitorio de un grafo), es posible determinar los estados que son alcanzables y los que no. Esta técnica se denomina **análisis de asequibilidad** (Lin y cols., 1987). Este análisis puede ser útil para determinar si un protocolo es correcto o no.

Formalmente, un modelo de máquina de estados finitos de un protocolo se puede considerar como un cuádruple (*S, M, I, T*), donde:

S es el conjunto de estados en que pueden estar los procesos y el canal.

M es el conjunto de tramas que pueden intercambiarse a través del canal.

I es el conjunto de estados iniciales de los procesos.

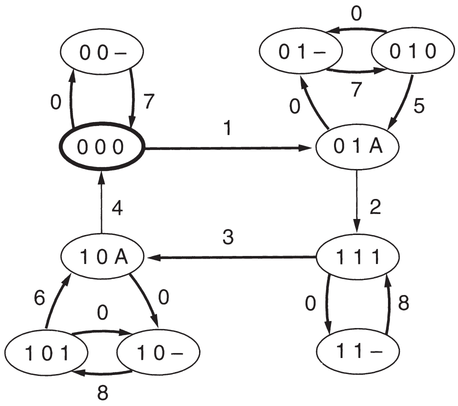
T es el conjunto de transiciones entre los estados.

Al principio todos los procesos están en sus estados iniciales. Entonces comienzan a ocurrir eventos, como tramas que se vuelven disponibles para transmisión o temporizadores que expiran. Cada evento puede causar que uno de los procesos o el canal emprenda una acción y cambie a un estado nuevo. Enumerando cuidadosamente cada sucesor posible para cada estado, podemos construir el grafo de asequibilidad y analizar el protocolo.

El análisis de asequibilidad puede servir para detectar una variedad de errores en la especificación del protocolo. Por ejemplo, si es posible que ocurra cierta trama en cierto estado y la máquina de estados finitos no indica la acción a tomar, la especificación es errónea (está incompleta). Si existe un grupo de estados para los que no hay salida y de los que no se puede avanzar (es decir, ya no es posible recibir tramas correctas), tenemos otro error (bloqueo irreversible). Un error menos grave es una especificación de protocolo que indica la manera de manejar un evento en un estado en que el evento no puede ocurrir (transición ajena). También pueden detectarse otros errores.

Como ejemplo de modelo de máquina de estados finitos, considere la figura 3-21(a). Este grafo corresponde al protocolo 3, como se describió antes: cada máquina de protocolo tiene dos estados y el canal tiene cuatro. Hay en total 16 estados, no todos alcanzables desde el inicial. Los inalcanzables no se muestran en la figura. Los errores de suma de verificación se ignoran aquí por simplicidad.

Se utilizan tres caracteres para etiquetar cada estado, *SRC*, donde *S* es 0 o 1 y corresponde a la trama que el emisor está tratando de enviar; *R* también es 0 o 1 y corresponde a la trama que el receptor espera, y *C* es 0, 1, *A* o vacío (−) y corresponde al estado del canal. En este ejemplo, el estado inicial se ha elegido como (000). En otras palabras, el emisor sólo ha enviado la trama 0, el receptor espera la trama 0 y ésta está actualmente en el canal.



(a)

Transición	¿Quién opera?	Trama aceptada	Trama emitida	A capa de red
0	–	(trama perdida)		–
1	R	0	A	Sí
2	S	A	1	–
3	R	1	A	Sí
4	S	A	0	–
5	R	0	A	No
6	R	1	A	No
7	S	(expiración de temporizador)	0	–
8	S	(expiración de temporizador)	1	–

(b)

Figura 3-21. (a) Diagrama de estado para el protocolo 3. (b) Transiciones.

En la figura 3-21 se muestran nueve tipos de transiciones. La transición 0 consiste en la pérdida del contenido del canal. La transición 1 consiste en que el canal entregue de manera correcta el paquete 0 al receptor, y que éste cambie a continuación su estado para esperar la trama 1 y emitir una confirmación de recepción. La transición 1 también tiene que ver con que el receptor entregue el paquete 0 a la capa de red. Las otras transiciones se listan en la figura 3-21(b). La llegada de una trama con un error de suma de verificación no se ha mostrado, pues no cambia el estado (en el protocolo 3).

Durante la operación normal, las transiciones 1, 2, 3 y 4 se repiten en orden una y otra vez. En cada ciclo se entregan dos paquetes, regresando el emisor al estado original de tratar de enviar una trama nueva con un número de secuencia 0. Si el canal pierde la trama 0, hace una transición del estado (000) al estado (00–). En algún momento, el temporizador del emisor expira (transición 7) y el sistema regresa a (000). La pérdida de una confirmación de recepción es más complicada, pues requiere dos transiciones, 7 y 5 u 8 y 6, para reparar el daño.

Una de las propiedades que debe tener un protocolo con un número de secuencia de 1 bit es que, sin importar la secuencia de eventos que ocurra, el receptor nunca debe entregar dos paquetes impares sin haber intervenido un paquete par, y viceversa. En el grafo de la figura 3-21 podemos ver que este requisito puede establecerse más formalmente como “no debe existir ninguna ruta desde el estado inicial en la que sucedan dos transiciones 1 sin que ocurra una transición 3 entre ellas, o al revés”. En la figura se puede ver que el protocolo es correcto en este aspecto.

Otro requisito semejante es que no debe haber rutas en las que el emisor pueda cambiar de estado dos veces (por ejemplo, de 0 a 1 y de regreso a 0) mientras el estado del receptor permanezca constante. De existir tal ruta, en la secuencia correspondiente de eventos se perderían dos tramas sin posibilidad de recuperación y sin que el receptor se diera cuenta. La secuencia de paquetes entregados tendrá un hueco no detectado de dos paquetes.

Otra propiedad importante de un protocolo es la ausencia de bloqueos irreversibles. Un **bloqueo irreversible** es una situación en la que el protocolo no puede seguir avanzando (es decir, entregando paquetes a la capa de red), sea cual sea la secuencia de eventos que ocurra. En términos del modelo gráfico, un bloqueo irreversible se caracteriza por la existencia de un subconjunto de estados que es alcanzable desde el estado inicial y que tiene dos propiedades:

1. No hay transición hacia fuera del subconjunto.
2. No hay transiciones en el subconjunto que causen un avance.

Una vez en el estado de bloqueo irreversible, el protocolo permanece ahí eternamente. De nuevo, es fácil ver en el grafo que el protocolo 3 no sufre de bloqueos irreversibles.

3.5.2 Modelos de red de Petri

La máquina de estados finitos no es la única técnica para especificar protocolos formalmente. En esta sección describiremos una técnica completamente diferente, la **red de Petri** (Danthine, 1980). Una red de Petri tiene cuatro elementos básicos: lugares, transiciones, arcos y *tokens*. Un **lugar** representa un estado en el que puede estar parte del sistema. En la figura 3-22 se muestra una red de Petri con dos lugares, *A* y *B*, que aparecen como círculos. El sistema actualmente está en el estado *A*, indicado por el **token** (punto grueso) en el lugar *A*. Se utiliza una barra horizontal o vertical para indicar una **transición.** Cada transición tiene cero o más **arcos de entrada,** que llegan de sus lugares de entrada, y cero o más **arcos de salida,** que van a sus lugares de salida.

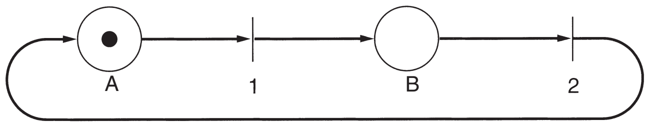


Figura 3-22. Red de Petri con dos lugares y dos transiciones.

Se **habilita** una transición si hay cuando menos un *token* de entrada en cada uno de sus lugares de entrada. Cualquier transición habilitada puede **dispararse** a voluntad, quitando un *token* de cada lugar de entrada y depositando un *token* en cada lugar de salida. Si el número de arcos de entrada no es igual al número de arcos de salida, no se conservarán los *tokens*. Si se habilitan dos o más transiciones, cualquiera de ellas puede dispararse. La decisión de disparo de una transición es indeterminada, por lo que las redes de Petri son útiles para modelar protocolos. La red de Petri de la figura 3-22 es determinista y puede servir para modelar cualquier proceso de dos fases (por ejemplo, el comportamiento de un bebé: comer, dormir, comer, dormir, y así sucesivamente). Como con todas las herramientas de modelado, se omiten los detalles innecesarios.

En la figura 3-23 se presenta el modelo de red de Petri de la figura 3-22. A diferencia del modelo de máquina de estados finitos, aquí no hay estados compuestos; el estado del emisor, el estado del canal y el estado del receptor se representan por separado. Las transiciones 1 y 2 corresponden al

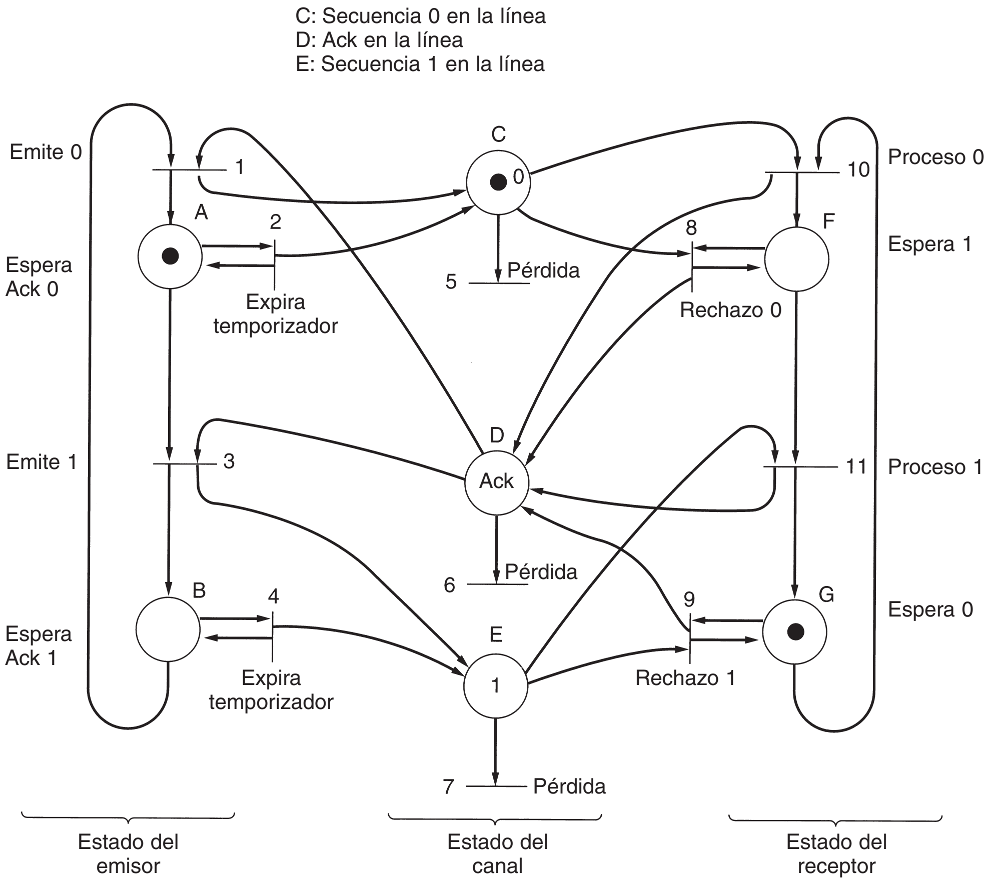


Figura 3-23. Modelo de red de Petri para el protocolo 3.

envío de la trama 0 por el emisor, normalmente, y al momento de una expiración de temporizador, respectivamente. Las transiciones 3 y 4 son análogas para la trama 1. Las transiciones 5, 6 y 7 corresponden a la pérdida de la trama 0, de una confirmación de recepción y de la trama 1, respectivamente. Las transiciones 8 y 9 ocurren cuando una trama de datos con un número de secuencia equivocado llega al receptor. Las transiciones 10 y 11 representan la llegada al receptor de la siguiente trama en la secuencia y su entrega a la capa de red.

Las redes de Petri pueden servir para detectar fallas de protocolo de una manera parecida a como se hace con máquinas de estados finitos. Por ejemplo, si alguna secuencia de disparo incluyera la transición 10 dos veces sin interponerse la transición 11, el protocolo sería incorrecto. El concepto de bloqueo irreversible en una red de Petri es semejante a su contraparte en una máquina de estados finitos.

Las redes de Petri pueden representarse convenientemente en una forma algebraica semejante a una gramática. Cada transición contribuye con una regla a la gramática. Cada regla especifica lugares de entrada y salida de la transición. Debido a que la figura 3-23 tiene 11 transiciones, su

gramática tiene 11 reglas, numeradas del 1 al 11, y cada una corresponde a la transición del mismo número. La gramática de la red de Petri de la figura 3-23 es como se muestra a continuación:

1: BD → AC
2: A → A
3: AD → BE
4: B → B
5: C →
6: D →
7: E →
8: CF → DF
9: EG → DG
10: CG → DF
11: EF → DG

Es interesante mencionar cómo hemos reducido un protocolo complejo a 11 reglas simples de gramática que pueden ser manipuladas fácilmente por un programa de computadora.

El estado actual de la red de Petri se representa como una colección desordenada de lugares, cada uno representado en la colección dependiendo de los *tokens* que tenga. Cualquier regla con lugares presentes en su izquierda puede ser disparada, removiendo aquellos lugares de su estado actual y agregando sus lugares de salida al estado actual. El marcado de la figura 3-23 es *ACG* (es decir, tanto *A*, como *C* y *G* tienen un *token*). En consecuencia, las reglas 2, 5 y 10 están habilitadas y cualquiera de ellas puede aplicarse, lo que lleva a un nuevo estado (posiblemente con el mismo marcado que el original). En contraste, la regla 3 (*AD* → *BE*) no puede aplicarse porque *D* no está marcada.

3.6 EJEMPLOS DE PROTOCOLOS DE ENLACE DE DATOS

En las siguientes secciones examinaremos varios protocolos de enlace de datos de amplio uso. El primero, HDLC, es un protocolo clásico orientado a bits cuyas variantes se han utilizado durante décadas en muchas aplicaciones. El segundo, PPP, es un protocolo de enlace utilizado para conectar a Internet computadoras domésticas.

3.6.1 HDLC—Control de Enlace de Datos de Alto Nivel

En esta sección examinaremos un grupo de protocolos íntimamente relacionados que, a pesar de ser un poco antiguos, se siguen utilizando ampliamente en redes de todo el mundo. Todas se derivan del primer protocolo de enlace de datos usado en los *mainframes* de IBM: el protocolo **SDLC (Control Síncrono de Enlace de Datos).** Después de desarrollar SDLC, IBM lo sometió al ANSI y a la ISO para su aceptación como estándar de Estados Unidos e internacional, respectivamente. El ANSI lo modificó convirtiéndolo en **ADCCP (Procedimiento Avanzado de Control de Comunicación de Datos),** y la ISO lo modificó para convertirlo en **HDLC (Control de Enlace de Datos de Alto Nivel).** Luego, el CCITT adoptó y modificó HDLC para su **LAP (Procedimiento de Acceso al Enlace)** como parte del estándar de interfaz de red X.25, pero después lo modificó nuevamente a **LAPB** para hacerlo más compatible con una versión posterior de

HDLC. Lo agradable de los estándares es que hay muchos de donde escoger. Es más, si no le gusta ninguno de ellos, simplemente puede sentarse a esperar el modelo del año próximo.

Todos estos protocolo se basan en el mismo principio. Todos son orientados a bits y usan el relleno de bits para lograr la transparencia de los datos. Difieren sólo en aspectos menores, aunque irritantes. El análisis de protocolos orientados a bits que haremos a continuación pretende ser una introducción general. Si desea detalles específicos de cualquier protocolo, consulte la definición adecuada.

Todos los protocolos orientados a bits utilizan la estructura de trama mostrada en la figura 3-24. El campo de *Dirección* es de importancia primordial en las líneas con múltiples terminales, pues sirve para identificar una de las terminales. En líneas punto a punto a veces se usan para distinguir los comandos de las respuestas.

Bits	8	8	8	≥0	16	8
	0 1 1 1 1 1 1 0	Dirección	Control	Datos	Suma de verificación	0 1 1 1 1 1 1 0

Figura 3-24. Formato de trama para protocolos orientados a bits.

El campo de *Control* se utiliza para números de secuencia, confirmaciones de recepción y otros propósitos, como se explicará más adelante.

El campo de *Datos* puede contener cualquier información y puede tener una longitud arbitraria, aunque la eficiencia de la suma de verificación disminuye conforme el tamaño de la trama aumenta, debido a la mayor probabilidad de múltiples errores en ráfaga.

El campo de *Suma de verificación* es un código de redundancia cíclica que utiliza la técnica que examinamos en la sección 3.2.2.

La trama está delimitada por otra secuencia de bandera (01111110). En líneas punto a punto inactivas se transmiten secuencias de bandera continuamente. La trama mínima contiene tres campos y un total de 32 bits, y excluye a las banderas de ambos lados.

Hay tres tipos de tramas: **de información, de supervisión** y **no numeradas.** El contenido del campo de *Control* para estos tres tipos se muestra en la figura 3-25. El protocolo emplea una ventana corrediza, con un número de secuencia de 3 bits. En cualquier momento pueden estar pendientes hasta siete tramas sin confirmación de recepción. El campo *Secuencia* de la figura 3-25(a) es el número de secuencia de la trama. El campo *Siguiente* es una confirmación de recepción superpuesta. Sin embargo, todos los protocolos se apegan a la convención de que, en lugar de superponer el número de la última trama recibida correctamente, usan el número de la primera trama no recibida (es decir, la siguiente trama esperada). La decisión de utilizar la última trama recibida o la siguiente trama esperada es arbitraria; no importa la convención que se utilice, siempre y cuando se use con consistencia.

El bit *P/F (S/F)* significa *Sondeo/Final (Poll/Final).* Se utiliza cuando una computadora (o concentrador) está sondeado un grupo de terminales. Cuando se usa como *P,* la computadora está invitando a la terminal a enviar datos. Todas las tramas enviadas por la terminal, excepto la última, tienen el bit *P/F* establecido en *P*. El último se establece en *F.*

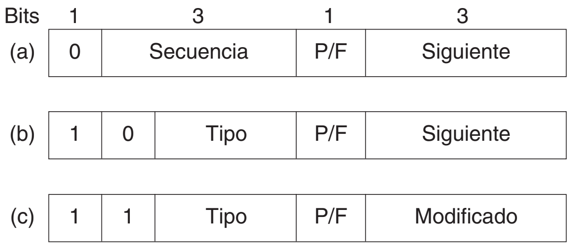


Figura 3-25. Campo de Control de (a) una trama de información, (b) una trama de supervisión y (c) una trama no numerada.

En algunos de los protocolos el bit *P/F* sirve para obligar a la otra máquina a enviar de inmediato una trama de supervisión, en lugar de esperar tráfico de regreso al cual superponer la información de la ventana. El bit también tiene algunos usos menores en conexión con las tramas sin número.

Los diferentes tipos de tramas de supervisión se distinguen por el campo de *Tipo.* El tipo 0 es una trama de confirmación de recepción (oficialmente llamada RECEIVE READY) que sirve para indicar la siguiente trama esperada. Ésta se usa cuando no hay tráfico de regreso que se pueda aprovechar para superponer confirmaciones de recepción.

El tipo 1 es una trama de confirmación de recepción negativa (oficialmente llamada REJECT); sirve para indicar que se ha detectado un error de transmisión. El campo *siguiente* indica la primera trama en la secuencia que no se ha recibido en forma correcta (es decir, la trama a retransmitir). Se pide al emisor retransmitir todas las tramas pendientes comenzando por *siguiente*. Esta estrategia es semejante a la de nuestro protocolo 5, más que a la de nuestro protocolo 6.

El tipo 2 es RECEIVE NOT READY (receptor no listo); reconoce todas las tramas hasta, pero sin incluir, *siguiente*, al igual que RECEIVE NOT READY, pero le dice al emisor que detenga el envío. El propósito de RECEIVE NOT READY es señalar ciertos problemas temporales en el receptor, como falta de búfer, y no servir como alternativa del control de flujo de ventana corrediza. Cuando el problema se resuelve, el receptor envía RECEIVE READY, REJECT o ciertas tramas de control.

El tipo 3 es SELECTIVE REJECT; solicita la retransmisión de sólo la trama especificada. En este sentido es como nuestro protocolo 6, no como el 5, y por lo tanto es de mayor utilidad cuando el tamaño de la ventana del emisor es la mitad del tamaño del espacio secuencial, o menor. Por lo tanto, si un receptor desea colocar en búferes tramas fuera de secuencia para un potencial uso futuro, puede reforzar la retransmisión de cualquier trama específica usando SELECTIVE REJECT. HDLC y ADCCP permiten este tipo de tramas, pero SDLC y LAPB no lo permiten (es decir, no hay rechazo selectivo), y las tramas de tipo 3 no están definidas.

La tercera clase de trama es la trama no mumerada que a veces se usa para propósitos de control, aunque también puede servir para llevar datos cuando se solicita un servicio no confiable sin conexión. Los diferentes protocolos orientados a bits tienen diferencias considerables aquí, en contraste con los otros dos tipos, donde son casi idénticos. Hay cinco bits disponibles para indicar el tipo de trama enviada, pero no se utilizan las 32 posibilidades.

Todos los protocolos proporcionan un comando, DISC (*DISConnect*), que permite a una máquina anunciar que va a ser desactivada (por ejemplo, para mantenimiento preventivo). También cuentan con un comando que permite a una máquina que acaba de regresar y está en línea anunciar su presencia y obligar el regreso a cero de todos los números de secuencia. Este comando se llama SNRM (Establecer Modo de Respuesta Normal). Desgraciadamente, el "modo de respuesta normal" es todo menos normal. Es un modo desbalanceado (es decir, asimétrico) en el que un extremo de la línea es el maestro y el otro, el subordinado. SNRM se remonta a una época en la que "comunicación de datos" significaba una terminal no inteligente que se comunicaba con una enorme computadora *host*, lo cual evidentemente es asimétrico. Para hacer más adecuado el protocolo cuando los dos interlocutores son iguales, HDLC y LAPB cuentan con un comando adicional, SABM (Establecer Modo Asíncrono Balanceado), que reestablece la línea y declara que ambas partes son iguales. También cuentan con los comandos SABME y SNRME, que son iguales a SABM y a SNRM, respectivamente, excepto que habilitan un formato de trama extendida que maneja números de secuencia de 7 bits en lugar de 3 bits.

Un tercer comando proporcionado por todos los protocolos es FRMR (Rechazo de Trama), que sirve para indicar que ha llegado una trama con suma de verificación correcta pero semántica imposible. Ejemplos de semántica imposible son: una trama de supervisión tipo 3 en LAPB, una trama menor a 32 bits, una trama de control ilegal, la confirmación de recepción de una trama que estaba fuera de la ventana, etcétera. Las tramas FRMR contienen un campo de datos de 24 bits que indica por qué se rechazó la trama. Los datos incluyen el campo de control de la trama rechazada, los parámetros de la ventana y un conjunto de bits que señalan errores específicos.

Las tramas de control pueden perderse o dañarse, igual que las de datos, por lo que también se debe confirmar su recepción. Se proporciona una trama de control especial para este propósito, llamada UA (Confirmación de Recepción no Numerada). Debido a que sólo puede estar pendiente una trama de control, nunca hay ambigüedades sobre la trama de control que está siendo confirmada.

Las tramas de control restantes tienen que ver con la inicialización, sondeo e informe de estado. También hay una trama de control que puede contener información arbitraria, UI (Información no Numerada). Estos datos no se pasan a la capa de red, pues son para uso de la capa de enlace de datos.

A pesar de su uso extendido, HDLC está lejos de ser perfecto. En (Fiorini y cols., 1994) puede encontrar un análisis de los distintos problemas asociados a este protocolo.

3.6.2 La capa de enlace de datos en Internet

Internet consiste en máquinas individuales (*hosts* y enrutadores) y la infraestructura de comunicación que las conecta. Dentro de un solo edificio, las LANs se usan ampliamente para la interconexión, pero la mayor parte de la infraestructura de área amplia está construida a partir de líneas alquiladas punto a punto. En el capítulo 4 veremos las LANs; aquí examinaremos los protocolos de enlace de datos usados en las líneas punto a punto de Internet.

En la práctica, la comunicación punto a punto se utiliza principalmente en dos situaciones. Primero, miles de organizaciones tienen una o más LANs, cada una con cierta cantidad de *hosts*

(computadoras personales, estaciones de trabajo, servidores y otros) junto con un enrutador (o un puente, que funcionalmente es parecido). Con frecuencia, los enrutadores se interconectan mediante una LAN de red dorsal. Por lo general, todas las conexiones al mundo exterior pasan a través de uno o dos enrutadores que tienen líneas alquiladas punto a punto a enrutadores distantes. Son estos enrutadores y sus líneas arrendadas los que conforman las subredes de comunicación sobre las que está construida Internet.

La segunda situación en la que las líneas punto a punto desempeñan un papel principal en Internet son los millones de personas que tienen conexiones domésticas a Internet a través de módems y líneas de acceso telefónico. Generalmente lo que ocurre es que la PC doméstica del usuario llama a un enrutador del proveedor de servicios de Internet y luego actúa como *host* de Internet. Este método de operación no es diferente de tener una línea alquilada entre la PC y el enrutador, excepto que la conexión se termina cuando el usuario termina la sesión. En la figura 3-26 se ilustra una PC doméstica que llama a un proveedor de servicios de Internet. El módem que se muestra es externo a la computadora para enfatizar su papel, pero las computadoras modernas tienen módems internos.

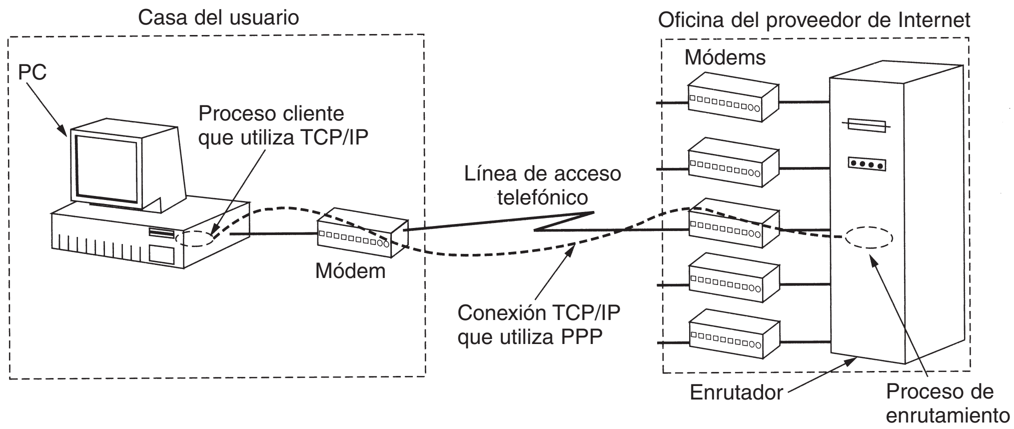


Figura 3-26. Computadora personal doméstica que funciona como *host* de Internet.

Tanto para la conexión por línea alquilada de enrutador a enrutador como para la conexión de acceso telefónico de *host* a enrutador, en la línea se requiere un protocolo de enlace de datos punto a punto para el entramado, el control de errores y las demás funciones de la capa de enlace de datos que hemos estudiado en este capítulo. El que se utiliza en Internet se conoce como PPP. A continuación lo analizaremos.

PPP—Protocolo Punto a Punto

Internet necesita de un protocolo punto a punto para diversos propósitos, entre ellos para el tráfico enrutador a enrutador y tráfico usuario doméstico a ISP. Este protocolo es **PPP (Protocolo**

Punto a Punto), que se define en el RFC 1661 y que se ha desarrollado más en varios otros RFCs (por ejemplo, los RFCs 1662 y 1663). PPP realiza detección de errores, soporta múltiples protocolos, permite la negociación de direcciones de IP en el momento de la conexión, permite la autenticación y tiene muchas otras funciones.

PPP proporciona tres características:

1. Un método de entramado que delinea sin ambigüedades el final de una trama y el inicio de la siguiente. El formato de trama también maneja la detección de errores.

2. Un protocolo de control de enlace para activar líneas, probarlas, negociar opciones y desactivarlas ordenadamente cuando ya no son necesarias. Este protocolo se llama **LCP (Protocolo de Control de Enlace).** Admite circuitos síncronos y asíncronos y codificaciones orientadas a bits y a caracteres.

3. Un mecanismo para negociar opciones de capa de red con independencia del protocolo de red usado. El método escogido consiste en tener un **NCP (Protocolo de Control de Red)** distinto para cada protocolo de capa de red soportado.

Para ver la manera en que encajan estas piezas, consideremos la situación típica de un usuario doméstico llamando al proveedor de servicios de Internet para convertir una PC doméstica en un *host* temporal de Internet. La PC llama inicialmente al enrutador del proveedor a través de un módem. Una vez que el módem del enrutador ha contestado el teléfono y ha establecido una conexión física, la PC manda al enrutador una serie de paquetes LCP en el campo de carga útil de una o más tramas PPP. Estos paquetes, y sus respuestas, seleccionan los parámetros PPP por usar.

Una vez que se han acordado estos parámetros, se envía una serie de paquetes NCP para configurar la capa de red. Por lo general, la PC requiere ejecutar una pila de protocolos TCP/IP, por lo que necesita una dirección IP. No hay suficientes direcciones IP para todos, por lo que normalmente cada proveedor de Internet tiene un bloque de ellas y asigna de manera dinámica una a cada PC que se acaba de conectar para que la use durante su sesión. Si un proveedor posee *n* direcciones IP, puede tener hasta *n* máquinas conectadas en forma simultánea, pero su base total de clientes puede ser muchas veces mayor. Se utiliza el NCP para IP para asignar la dirección IP.

En este momento, la PC es ya un *host* de Internet y puede enviar y recibir paquetes IP, igual que los *host* permanentes. Cuando el usuario ha terminado, se utiliza NCP para desmantelar la conexión de la capa de red y liberar la dirección IP. Luego se usa LCP para cancelar la conexión de la capa de enlace de datos. Por último, la computadora indica al módem que cuelgue el teléfono, liberando la conexión de la capa física.

El formato de trama de PPP se escogió de modo que fuera muy parecido al de HDLC, ya que no había razón para reinventar la rueda. La diferencia principal entre PPP y HDLC es que el primero está orientado a caracteres, no a bits. En particular, PPP usa el relleno de bytes en las líneas de acceso telefónico con módem, por lo que todas las tramas tienen un número entero de bytes. No es posible enviar una trama que conste de 30.25 bytes, como con HDLC. No sólo pueden mandarse tramas PPP a través de líneas de acceso telefónico, sino que también pueden enviarse a través

de SONET o de líneas HDLC auténticas orientadas a bits (por ejemplo, para conexiones enrutador a enrutador). El formato de trama PPP se muestra en la figura 3-27.

Bytes	1	1	1	1 o 2	Variable	2 o 4	1
	Bandera 01111110	Dirección 11111111	Control 00000011	Protocolo	Carga útil	Suma de verificación	Bandera 01111110

Figura 3-27. Formato de trama completa PPP para el modo de operación no numerado.

Todas las tramas PPP comienzan con la bandera estándar de HDLC (01111110), que se rellena con bytes si ocurre dentro del campo de carga útil. Luego está el campo de *Dirección,* que siempre se establece al valor binario 11111111 para indicar que todas las estaciones deben aceptar la trama. El empleo de este valor evita tener que asignar direcciones de la capa de enlace de datos.

El campo de *Dirección* va seguido del campo de *Control*, cuyo valor predeterminado es 00000011. Este valor indica una trama no numerada. En otras palabras, PPP no proporciona de manera predeterminada transmisión confiable usando números de secuencia y confirmaciones de recepción. En entornos ruidosos, como los de las redes inalámbricas, se puede emplear el modo numerado para transmisión confiable, aunque casi no se utiliza. Los detalles exactos se definen en el RFC 1663.

Dado que los campos de *Dirección* y de *Control* son constantes en la configuración predeterminada, LCP proporciona los mecanismos necesarios para que las dos partes negocien una opción que simplemente los omita por completo y ahorre dos bytes por trama.

El cuarto campo PPP es el de *Protocolo*. Su tarea es indicar la clase de paquete que está en el campo de *Carga útil*. Se definen códigos para LCP, NCP, IP, IPX, AppleTalk, entre otros. Los protocolos que comienzan con un bit 0 son de capa de red como IP, IPX, OSI CLNP, XNS. Los que comienzan con un bit 1 se utilizan para negociar otros protocolos. Entre éstos están LCP y un NCP diferente para cada protocolo de capa de red soportado. El tamaño predeterminado del campo de *protocolo* es de 2 bytes, pero puede negociarse a 1 byte usando LCP.

El campo de *Carga útil* es de longitud variable, hasta algún máximo negociado. Si la longitud no se negocia con LCP durante el establecimiento de la línea, se usa una longitud predeterminada de 1500 bytes. De ser necesario se puede incluir un relleno después de la carga.

Después del campo de *Carga útil* está el campo de *Suma de verificación*, que normalmente es de 2 bytes, pero puede negociarse una suma de verificación de 4 bytes.

En resumen, PPP es un mecanismo de entramado multiprotocolo adecuado para utilizarse a través de módems, líneas seriales de bits HDLC, SONET y otras capas físicas. Soporta detección de errores, negociación de opciones, compresión de encabezados y, opcionalmente, transmisión confiable con formato de tramas similar al de HDLC.

Ahora pasemos del formato de tramas PPP a la manera en que se activan y desactivan las líneas. En el diagrama (simplificado) de la figura 3-28 se muestran las fases por las que pasa una

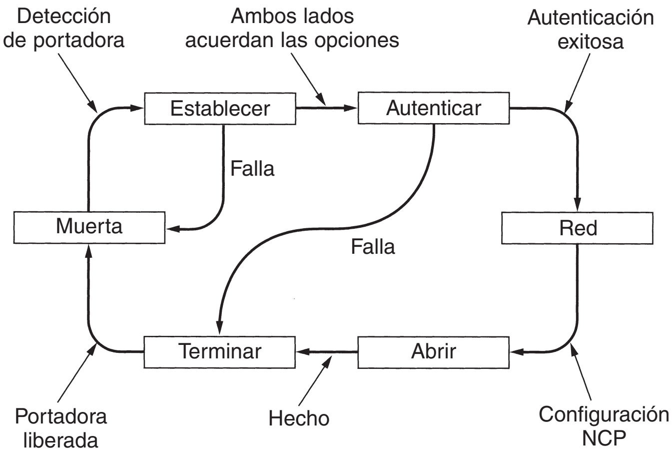


Figura 3-28. Diagrama de fases simplificado para activar y desactivar una línea.

línea cuando es activada, usada y desactivada. Esta secuencia se aplica tanto a las conexiones por módem como a las de enrutador a enrutador.

El protocolo inicia con la línea que tiene el estado *MUERTA*, lo que significa que no hay una portadora de capa física y que no existe una conexión de capa física. Una vez establecida la conexión física, la línea pasa a *ESTABLECER*. En ese punto comienza la negociación de opciones LCP, que, de tener éxito, conduce a *AUTENTICAR*. Ahora las dos partes pueden verificar la identidad del otro, si lo desean. Al entrar en la fase *RED*, se invoca el protocolo NCP apropiado para configurar la capa de red. Si la configuración tiene éxito, se llega a *ABRIR* y puede comenzar el transporte de datos. Al terminar el transporte, la línea pasa a la fase *TERMINAR*, de donde regresa a *MUERTA* al liberarse la portadora.

Se utiliza LCP para negociar las opciones del protocolo de enlace de datos durante la fase *ESTABLECER*. El protocolo LCP no se ocupa realmente de las opciones mismas, sino de los mecanismos de negociación; proporciona una vía para que el proceso iniciador haga una propuesta y para que el proceso que responde la acepte o la rechace, toda o en parte. También proporciona una forma para que los dos procesos prueben la calidad de la línea, y vean si es lo suficientemente buena para establecer una conexión. Por último, el protocolo LCP también permite que las líneas se desactiven cuando ya no se necesitan.

Hay 11 tipos de tramas LCP definidas en el RFC 1661, y se listan en la figura 3-29. Los cuatro tipos de *configuración* permiten que el iniciador (I) proponga valores de opción y que el contestador (R) las acepte o las rechace. En el último caso, el contestador puede hacer una propuesta alterna o anunciar que no está dispuesto a negociar en absoluto. Las opciones negociadas y sus valores propuestos son parte de las tramas LCP.

Los códigos de *terminación* sirven para desactivar una línea cuando ya no se necesita. El contestador utiliza los códigos de *código-rechazo* y *protocolo-rechazo* para indicar que recibió algo que no entiende. Esta situación puede significar que ha ocurrido un error de transmisión no detectado,

Nombre	Dirección	Descripción
Configurar-solicitud	I → R	Lista de opciones y valores propuestos
Configurar-confirmación de recepción	I ← R	Se aceptan todas las opciones
Configurar-confirmación de recepción negativa	I ← R	No se aceptan algunas opciones
Configurar-rechazo	I ← R	Algunas opciones no son negociables
Terminar-solicitud	I → R	Solicitud de desactivación de la línea
Terminar-confirmación de recepción	I ← R	Línea desactivada
Código-rechazo	I ← R	Se recibió una solicitud desconocida
Protocolo-rechazo	I ← R	Se solicitó un protocolo desconocido
Eco-solicitud	I → R	Favor de enviar de regreso esta trama
Eco-respuesta	I ← R	Aquí está la trama de regreso
Descartar-solicitud	I → R	Descartar esta trama (para pruebas)

Figura 3-29. Tipos de tramas LCP.

pero más probablemente significa que el iniciador y el contestador están ejecutando versiones diferentes del protocolo LCP. Los códigos *eco* sirven para probar la calidad de la línea. Por último, *descartar-solicitud* se utiliza para depuración. Si cualquiera de los lados está teniendo problemas para poner bits en la línea (transmitir), el programador puede emplear este tipo para pruebas. Si logra pasar, el receptor simplemente lo descarta en lugar de realizar alguna acción que podría confundir a la persona que efectúa las pruebas.

Las opciones que pueden negociarse incluyen el establecimiento del tamaño máximo de la carga útil para las tramas de datos, la habilitación de la autenticación y la selección del protocolo a emplear, habilitando el monitoreo de la calidad de la línea durante la operación normal y la selección de varias opciones de compresión de encabezados.

Hay poco que decir sobre los protocolos NCP en un sentido general. Cada uno es específico para algún protocolo de capa de red y permite que se hagan solicitudes de configuración específicas de ese protocolo. Por ejemplo, para IP la posibilidad más importante es la asignación dinámica de direcciones.

3.7 RESUMEN

La tarea de la capa de enlace de datos es convertir el flujo de bits en bruto ofrecido por la capa física en un flujo de tramas para que la capa de red lo utilice. Se emplean varios métodos de entramado, incluidos el conteo de caracteres, el relleno de bytes y el relleno de bits. Los protocolos de enlace de datos pueden proporcionar control de errores para retransmitir tramas dañadas o perdidas. Para evitar que un emisor rápido sature a un receptor lento, el protocolo de enlace de datos también puede proporcionar control de flujo. El mecanismo de ventana corrediza se emplea ampliamente para integrar el control de errores y el control de flujo de una manera conveniente.

Los protocolos de ventana corrediza pueden clasificarse por el tamaño de la ventana del emisor y por el tamaño de la ventana del receptor. Cuando ambos son iguales a 1, el protocolo es de parada y espera. Cuando el tamaño de la ventana del emisor es mayor que 1, por ejemplo, para evitar el bloqueo del emisor en un circuito con un retardo de propagación grande, el receptor puede programarse para descartar todas las tramas diferentes a la siguiente de la secuencia o almacenar en el búfer tramas fuera de orden hasta que se necesiten.

En este capítulo examinamos una serie de protocolos. El protocolo 1 se designa para un entorno libre de errores, en el que el receptor puede manejar cualquier flujo que se le haya enviado. El protocolo 2 también da por hecho un entorno libre de errores pero introduce control de flujo. El protocolo 3 maneja los errores con números de secuencia y con el algoritmo de parada y espera. El protocolo 4 permite la comunicación bidireccional e introduce el concepto de superposición. El protocolo 5 utiliza un protocolo de ventana corrediza con retroceso n. Por último, el protocolo 6 utiliza repetición selectiva y confirmaciones de recepción negativas.

Los protocolos pueden modelarse usando varias técnicas para demostrar que son correctos (o no). Los modelos de máquina de estados finitos y de red de Petri se usan frecuentemente para este propósito.

Muchas redes utilizan uno de los protocolos orientados a bits (SDLC, HDLC, ADCCP o LAPB) en la capa de enlace de datos. Todos estos protocolos usan bytes de bandera para delimitar tramas y relleno de bits para evitar que los bytes de bandera ocurran en los datos. Todos ellos también usan ventanas corredizas para el control de flujo. Internet utiliza PPP como el protocolo de enlace de datos primario en líneas punto a punto.

PROBLEMAS

1. Un mensaje de capa superior se divide en 10 tramas, cada una de las cuales tiene 80% de probabilidad de llegar sin daño. Si el protocolo de enlace de datos no lleva a cabo control de errores, ¿cuántas veces debe reenviarse el mensaje en promedio para conseguir que pase todo?

2. La siguiente codificación de caracteres se utiliza en un protocolo de enlace de datos:
   A: 01000111; B: 11100011; FLAG: 01111110; ESC: 11100000
   Muestre la secuencia de bits transmitida (en binario) para la trama de cuatro caracteres: A B ESC FLAG cuando se utiliza cada uno de los siguientes métodos de entramado:
   (a) Conteo de caracteres.
   (b) Bytes de bandera con relleno de bytes.
   (c) Bytes de bandera de inicio y final, con relleno de bits.

3. El siguiente fragmento de datos ocurre a la mitad de un flujo de datos para el cual se ha usado el algoritmo de relleno de bytes descrito en el texto: A B ESC C ESC FLAG FLAG D. ¿Cuál es la salida tras el relleno?

4. Uno de sus compañeros ha señalado que es un desperdicio terminar cada trama con un byte de bandera e iniciar la siguiente con otro. Un solo byte de bandera podría hacer el trabajo, por lo que un byte guardado es un byte ganado. ¿Usted está de acuerdo?

5. Una cadena de bits, 0111101111101111110, necesita transmitirse en la capa de enlace de datos. ¿Cuál es la cadena que realmente se está transmitiendo después del relleno de bits?

6. Cuando se usa relleno de bits, ¿es posible que la pérdida, inserción o modificación de un solo bit cause un error que la suma de verificación no detecte? Si no, ¿por qué no? Si es así, explique cómo. ¿Desempeña aquí un papel la longitud de la suma de verificación?

7. ¿Puede pensar en alguna circunstancia en la cual podría ser preferible un protocolo de ciclo abierto (por ejemplo, un código de Hamming) a los protocolos tipo realimentación analizados a lo largo de este capítulo?

8. Para proporcionar mayor confiabilidad de la que puede dar un solo bit de paridad, un esquema de codificación de detección de errores usa un bit de paridad para todos los bits de número par. ¿Cuál es la distancia de Hamming de este código?

9. Se utiliza el código de Hamming para transmitir mensajes de 16 bits. ¿Cuántos bits de verificación se necesitan para asegurar que el receptor pueda detectar y corregir errores de un solo bit? Muestre el patrón de bits transmitido para el mensaje 1101001100110101. Suponga que se utiliza paridad par en el código de Hamming.

10. Un byte de 8 bits con un valor binario de 10101111 se va a codificar utilizando código de Hamming de paridad par. ¿Cuál es el valor binario que resulta de la codificación?

11. Un código de Hamming de 12 bits, cuyo valor hexadecimal es 0xE4F, llega al receptor. ¿Cuál era el valor hexadecimal original? Suponga que no más de un bit es erróneo.

12. Una manera de detectar errores es transmitir los datos como un bloque de n filas de k bits por fila y agregar bits de paridad a cada fila y a cada columna. La esquina inferior derecha es un bit de paridad que verifica su fila y su columna. ¿Detectará este esquema todos los errores sencillos? ¿Los errores dobles? ¿Los errores triples?

13. Un bloque de bits con n filas y k columnas usa bits de paridad horizontales y verticales para la detección de errores. Suponga que se invierten exactamente 4 bits debido a errores de transmisión. Deduzca una expresión para la probabilidad de que el error no sea detectado.

14. ¿Qué residuo se obtiene al dividir $x^7 + x^5 + 1$ entre el polinomio generador $x^3 + 1$?

15. Un flujo de bits 10011101 se transmite utilizando el método estándar CRC que se describió en el texto. El generador polinomial es $x^3 + 1$. Muestre la cadena de bits real que se transmite. Suponga que el tercer bit, de izquierda a derecha, se invierte durante la transmisión. Muestre que este error se detecta en el lado receptor.

16. Los protocolos de enlace de datos casi siempre ponen el CRC en un terminador, en lugar de un encabezado. ¿Por qué?

17. Un canal tiene una tasa de bits de 4 kbps y un retardo de propagación de 20 mseg. ¿Para qué intervalo de tamaños de trama, la parada y espera da una eficiencia de cuando menos 50%?

18. Una troncal T1 de 3000 km de longitud se usa para transmitir tramas de 64 bytes con el protocolo 5. Si la velocidad de propagación es de 6 μseg/km, ¿de cuántos bits deben ser los números de secuencia?

19. En el protocolo 3, ¿es posible que el emisor inicie el temporizador cuando éste ya está en ejecución? De ser así, ¿cómo podría ocurrir? De lo contrario, ¿por qué no es posible?

20. Imagine un protocolo de ventana corrediza que utiliza tantos bits para los números de secuencia que nunca ocurre un reinicio. ¿Qué relaciones deben mantenerse entre los cuatro límites de la ventana y el tamaño de la ventana, que es constante y el mismo tanto para el emisor como para el receptor?

21. Si el procedimiento *between* del protocolo 5 revisara la condición $a \le b \le c$ en lugar de la condición $a \le b < c$, ¿tendría esto algún efecto en la corrección o en la eficiencia del protocolo? Explique su respuesta.

22. En el protocolo 6, cuando llega una trama de datos, se hace una revisión para ver si el número de secuencia es diferente del esperado y si *no_nak* es verdadero. Si ambas condiciones se cumplen, se envía una NAK. De otra manera, se arranca el temporizador auxiliar. Suponga que se omite la cláusula `else`. ¿Afectará este cambio la corrección del protocolo?

23. Suponga que el ciclo `while` de tres instrucciones cerca del final del protocolo 6 se elimina del código. ¿Afectará esto la corrección del protocolo o sólo su desempeño? Explique su respuesta.

24. Suponga que el caso para errores de suma de verificación fuera eliminado de la instrucción `switch` del protocolo 6. ¿Cómo afectará este cambio la operación del protocolo?

25. En el protocolo 6, el código de *frame_arrival* tiene una sección que se usa para los NAKs. Dicha sección se invoca si la trama entrante es una NAK y se cumple otra condición. Describa un escenario en el que la presencia de esta otra condición sea esencial.

26. Imagine que está escribiendo el software de la capa de enlace de datos para una línea que se usa para enviar datos a usted, pero no desde usted. El otro extremo usa HDLC, con un número de secuencia de tres bits y un tamaño de ventana de siete tramas. Usted podría querer almacenar en el búfer tantas tramas fuera de secuencia como fuera posible para aumentar la eficiencia, pero no se le permite modificar el software del lado emisor. ¿Es posible tener una ventana de receptor mayor que uno y aun así garantizar que el protocolo nunca fallará? De ser así, ¿cuál es el tamaño de la ventana más grande que se puede usar con seguridad?

27. Considere la operación del protocolo 6 en una línea de 1 Mbps libre de errores. El tamaño máximo de trama es de 1000 bits. Se generan nuevos paquetes a intervalos aproximados de 1 segundo. El tiempo de expiración del temporizador es de 10 mseg. Si se eliminara el temporizador especial de confirmación de recepción ocurrirían terminaciones de temporizador innecesarias. ¿Cuántas veces se transmitiría el mensaje promedio?

28. En el protocolo 6, $MAX_SEQ = 2^n - 1$. Si bien esta condición es evidentemente deseable para utilizar de manera eficiente los bits de encabezado, no hemos demostrado que sea esencial. ¿Funciona correctamente el protocolo con $MAX_SEQ = 4$, por ejemplo?

29. Se están enviando tramas de 1000 bits a través de un canal de 1 Mbps utilizando un satélite geoestacionario cuyo tiempo de propagación desde la Tierra es de 270 mseg. Las confirmaciones de recepción siempre se superponen en las tramas de datos. Los encabezados son muy cortos. Se usan números de secuencia de tres bits. ¿Cuál es la utilización máxima de canal que se puede lograr para:
(a) Parada y espera?
(b) El protocolo 5?
(c) El protocolo 6?

30. Calcule la fracción del ancho de banda que se desperdicia en sobrecarga (encabezados y retransmisiones) para el protocolo 6 en un canal satelital de 50 kbps con carga pesada, usando tramas de datos consistentes en 40 bits de encabezado y 3960 bits de datos. Asuma que el tiempo de propagación de la señal

de la Tierra al satélite es de 270 mseg. Nunca ocurren tramas ACK. Las tramas NAK son de 40 bits. La tasa de errores de las tramas de datos es de 1%, y la tasa de errores para las tramas NAK es insignificante. Los números de secuencia son de 8 bits.

31. Considere un canal satelital de 64 kbps libre de errores que se usa para enviar tramas de datos de 512 bytes en una dirección y devolver confirmaciones de recepción muy cortas en la otra. ¿Cuál es la velocidad real de transporte máxima con tamaños de ventana de 1, 7, 15 y 127? El tiempo de propagación de la Tierra al satélite es de 270 mseg.

32. Un cable de 100 km de longitud opera con una tasa de datos T1. La velocidad de propagación del cable es 2/3 de la velocidad de la luz en el vacío. ¿Cuántos bits caben en el cable?

33. Suponga que modelamos el protocolo 4 mediante el modelo de máquina de estados finitos. ¿Cuántos estados existen para cada máquina? ¿Cuántos estados existen para el canal de comunicaciones? ¿Cuántos estados existen para todo el sistema (dos máquinas y el canal)? Ignore los errores de suma de verificación.

34. Dé la secuencia de activación para la red de Petri de la figura 3-23 correspondiente a la secuencia de estado (000), (01A), (01—), (010), (01A) de la figura 3-21. Explique con palabras lo que representa la secuencia.

35. Dadas las reglas de transición $AC \rightarrow B$, $B \rightarrow AC$, $CD \rightarrow E$ y $E \rightarrow CD$, dibuje la red de Petri descrita. A partir de esta red, dibuje el grafo de estado finito alcanzable desde el estado inicial *ACD*. ¿Qué concepto bien conocido modelan estas reglas de transición?

36. PPP se basa estrechamente en HDLC, que utiliza relleno de bits para prevenir que los bytes de bandera accidentales dentro de la carga útil causen confusión. Dé por lo menos una razón por la cual PPP utiliza relleno de bytes.

37. ¿Cuál es la sobrecarga mínima para enviar un paquete IP mediante PPP? Tome en cuenta sólo la sobrecarga introducida por el PPP mismo, no la del encabezado IP.

38. El objetivo de este ejercicio es implementar un mecanismo de detección de errores utilizando el algoritmo CRC estándar descrito en el texto. Escriba dos programas: el generador y el verificador. El programa generador lee de una entrada estándar un mensaje de *n* bits como una cadena de 0s y 1s perteneciente a una línea de texto ASCII. La segunda línea es el código polinomial de *k* bits, también en ASCII. Ésta envía a la salida estándar una línea de texto ASCII con $n + k$ 0s y 1s que representan el mensaje que se va a transmitir. Después envía el código polinomial, justo como lo lee. El programa verificador lee la salida del programa generador y envía a la salida un mensaje que indica si es correcto o no. Por último, escriba un programa, de alteración, que invierta un bit en la primera línea dependiendo de su argumento (el número de bits considerando al bit más a la izquierda como 1), pero que copie el resto de las dos líneas de manera correcta. Si escribe:

```
generator <file | verifier
```

debe ver que el mensaje es correcto, pero si escribe

```
generator <file | alter arg | verifier
```

deberá obtener el mensaje de error.

39. Escriba un programa que simule el comportamiento de una red de Petri. El programa deberá leer las reglas de transición, así como una lista de estados que correspondan a la capa de enlace de red que emite o acepta un nuevo paquete. A partir del estado inicial, también de leído, el programa deberá elegir transiciones habilitadas al azar y activarlas, y verificar si el *host* acepta alguna vez 2 paquetes sin que el otro *host* emita uno nuevo en el ínterin.

4

LA SUBCAPA DE CONTROL DE ACCESO AL MEDIO

Como mencionamos en el capítulo 1, las redes pueden dividirse en dos categorías: las que utilizan conexiones punto a punto y las que utilizan canales de difusión. Este capítulo trata las redes de difusión y sus protocolos.

En cualquier red de difusión, el asunto clave es la manera de determinar quién puede utilizar el canal cuando hay competencia por él. Para aclarar este punto, considere una llamada en conferencia en la que seis personas, en seis teléfonos diferentes, están conectadas de modo que cada una puede oír y hablar con todas las demás. Es muy probable que cuando una de ellas deje de hablar, dos o más comiencen a hacerlo a la misma vez, lo que conducirá al caos. En las reuniones cara a cara, el caos se evita por medios externos. Por ejemplo, en una reunión, la gente levanta la mano para solicitar permiso de hablar. Cuando únicamente hay un canal, determinar quién debería tener el turno es muy complicado. Se conocen muchos protocolos para resolver el problema y constituyen el propósito de este capítulo. En la literatura, los canales de difusión a veces se denominan **canales multiacceso** o **canales de acceso aleatorio.**

Los protocolos usados para determinar quién sigue en un canal multiacceso pertenecen a una subcapa de la capa de enlace de datos llamada subcapa **MAC** (**Control de Acceso al Medio**). La subcapa MAC tiene especial importancia en las LANs, casi todas las cuales usan un canal multiacceso como base de su comunicación. Las WANs, en contraste, usan enlaces punto a punto, excepto en las redes satelitales. Debido a que los canales multiacceso y las LANs están estrechamente relacionados, en este capítulo analizaremos las LANs en general, además de algunos aspectos que no son estrictamente parte de la subcapa MAC.

Desde el punto de vista técnico, la subcapa MAC es la parte inferior de la capa de enlace de datos, por lo que lógicamente debimos haberla estudiado antes de examinar los protocolos punto a punto en el capítulo 3. No obstante, para la mayor parte de la gente, la comprensión de los protocolos en los que intervienen muchas partes es más fácil una vez que se han entendido bien los protocolos de dos partes. Por esta razón nos hemos desviado de un orden de presentación estrictamente ascendente.

4.1 EL PROBLEMA DE ASIGNACIÓN DEL CANAL

El tema central de este capítulo es la forma de asignar un solo canal de difusión entre usuarios competidores. Primero veremos los esquemas estático y dinámico en general. Luego examinaremos varios algoritmos específicos.

4.1.1 Asignación estática de canal en LANs y MANs

La manera tradicional de asignar un solo canal, como una troncal telefónica, entre varios usuarios competidores es la FDM (Multiplexión por División de Frecuencia). Si hay N usuarios, el ancho de banda se divide en N partes de igual tamaño (vea la figura 2-31), y a cada usuario se le asigna una parte. Dado que cada usuario tiene una banda de frecuencia privada, no hay interferencia entre los usuarios. Cuando sólo hay una pequeña cantidad fija de usuarios, cada uno de los cuales tiene (en búfer) una carga de tráfico pesada (por ejemplo, las oficinas de conmutación de una empresa portadora), la FDM es un mecanismo de asignación sencillo y eficiente.

Sin embargo, cuando el número de emisores es grande y varía continuamente, o cuando el tráfico se hace en ráfagas, la FDM presenta algunos problemas. Si el espectro se divide en N regiones, y hay menos de N usuarios interesados en comunicarse actualmente, se desperdiciará una buena parte de espectro valioso. Si más de N usuarios quieren comunicarse, a algunos de ellos se les negará el permiso por falta de ancho de banda, aun cuando algunos de los usuarios que tengan asignada una banda de frecuencia apenas transmitan o reciban algo.

Sin embargo, aun suponiendo que el número de usuarios podría, de alguna manera, mantenerse constante en N, dividir el canal disponible en subcanales estáticos es inherentemente ineficiente. El problema básico es que, cuando algunos usuarios están inactivos, su ancho de banda simplemente se pierde. No lo están usando, y a nadie más se le permite usarlo. Es más, en casi todos los sistemas de cómputo el tráfico de datos se hace en ráfagas (son comunes las relaciones de tráfico pico a tráfico medio de 1000:1). En consecuencia, la mayoría de los canales estarán inactivos casi todo el tiempo.

El desempeño pobre de la FDM estática puede verse fácilmente mediante un cálculo sencillo de la teoría de colas. Comencemos por el retardo medio, T, de un canal de C bps de capacidad, con una tasa de llegada de λ tramas/seg, en el que cada trama tiene una longitud que se obtiene de una función exponencial de densidad de probabilidad con una media de $1/\mu$ bits/trama. Con estos parámetros, la tasa de llegada es de λ tramas/seg y la tasa de servicio es de

μC tramas/seg. A partir de la teoría de colas se puede mostrar que para tiempos de llegada y de servicio de Poisson,

$$T = \frac{1}{\mu C - \lambda}$$

Por ejemplo, si C es 100 Mbps, la longitud media de trama, $1/\mu$, es 10,000 bits y la tasa de llegada de tramas, λ, es 5000 tramas/seg, entonces $T = 200$ μseg. Observe que si ignoráramos el retardo de colas y sólo preguntáramos cuánto toma enviar una trama de 10,000 bits en una red de 100 Mbps, podríamos obtener la respuesta (incorrecta) de 100 μseg. El resultado sólo es válido cuando no hay competencia por el canal.

Ahora dividamos el canal en N subcanales independientes, cada uno con capacidad de C/N bps. La tasa media de entrada de cada uno de los subcanales ahora será de λ/N. Recalculando T, obtenemos:

$$T_{\mathrm{FDM}} = \frac{1}{\mu(C/N) - (\lambda/N)} = \frac{N}{\mu C - \lambda} = NT \tag{4-1}$$

El retardo medio al usar FDM es N veces peor que si todas las tramas se acomodaran mágicamente de algún modo en una gran cola central.

Precisamente los mismos argumentos que se aplican a la FDM se aplican a la TDM (Multiplexión por División de Tiempo). A cada usuario se le asigna cada N-ésima ranura de tiempo. Si un usuario no usa la ranura asignada, simplemente se desperdicia. Lo mismo se aplica si dividimos las redes físicamente. Utilizando otra vez el ejemplo anterior, si reemplazamos la red de 100 Mbps con 10 redes de 10 Mbps cada una y asignamos de manera estática un usuario a cada una de ellas, el retardo medio reduciría de 200 μseg a 2 mseg.

Ya que ninguno de los métodos tradicionales de asignación estática de canal funciona muy bien con tráfico en ráfagas, ahora exploraremos los métodos dinámicos.

4.1.2 Asignación dinámica de canales en LANs y MANs

Antes de entrar en el primero de muchos métodos de asignación de canal que se analizarán en este capítulo, vale la pena formular con cuidado el problema de la asignación. Todo el trabajo hecho en esta área se basa en cinco supuestos clave, que se describen a continuación.

1. **Modelo de estación.** El modelo consiste en N **estaciones** independientes (computadoras, teléfonos, comunicadores personales, etcétera), cada una con un programa o usuario que genera tramas para transmisión. Algunas veces, las estaciones se conocen como **terminales.** La probabilidad de que una trama se genere en un intervalo de longitud Δt es de $\lambda \Delta t$, donde λ es una constante (la tasa de llegada de tramas nuevas). Una vez que se ha generado una trama, la estación se bloquea y no hace nada sino hasta que la trama se ha transmitido con éxito.

2. **Supuesto de canal único.** Hay un solo canal disponible para todas las comunicaciones. Todas las estaciones pueden transmitir en él y pueden recibir de él. En lo referente al hardware, todas las estaciones son equivalentes, aunque el software del protocolo puede asignarles prioridades.

3. **Supuesto de colisión.** Si dos tramas se transmiten en forma simultánea, se traslapan en el tiempo y la señal resultante se altera. Este evento se llama **colisión.** Todas las estaciones pueden detectar colisiones. Una trama en colisión debe transmitirse nuevamente después. No hay otros errores excepto aquellos generados por las colisiones.

4a. **Tiempo continuo.** La transmisión de una trama puede comenzar en cualquier momento. No hay reloj maestro que divida el tiempo en intervalos discretos.

4b. **Tiempo ranurado.** El tiempo se divide en intervalos discretos (ranuras). La transmisión de las tramas siempre comienza al inicio de una ranura. Una ranura puede contener 0, 1 o más tramas, correspondientes a una ranura inactiva, una transmisión con éxito o una colisión, respectivamente.

5a. **Detección de portadora.** Las estaciones pueden saber si el canal está en uso antes de intentar usarlo. Si se detecta que el canal está en uso, ninguna estación intentará utilizarlo sino hasta que regrese a la inactividad.

5b. **Sin detección de portadora.** Las estaciones no pueden detectar el canal antes de intentar usarlo. Simplemente transmiten. Sólo después pueden determinar si la transmisión tuvo éxito.

Es importante un análisis de estos supuestos. El primero dice que las estaciones son independientes, y que se genera trabajo a velocidad constante. También supone de manera implícita que cada estación sólo tiene un programa o usuario, así que mientras la estación esté bloqueada no se generará trabajo nuevo. Los modelos más complicados permiten estaciones multiprogramadas que pueden generar trabajo mientras la estación está bloqueada, pero el análisis de estas estaciones es mucho más complejo.

El supuesto del canal único es la esencia del modelo. No hay formas externas de comunicación. Las estaciones no pueden levantar la mano para solicitar que el maestro les ceda la palabra.

El supuesto de colisión también es básico, aunque en algunos sistemas (principalmente los de espectro disperso) este supuesto se suaviza con resultados sorprendentes. Además algunas LANs, como las *token ring*, pasan un *token* especial de estación en estación, y quien lo posea es quien puede transmitir una trama. Sin embargo, en las siguientes secciones nos apegaremos al modelo de canal único con contención y colisiones.

Hay dos supuestos alternativos sobre el tiempo. O es continuo (4a) o ranurado (4b). Algunos sistemas usan uno y otros el otro, por lo que estudiaremos y analizaremos ambos. Obviamente, para un sistema dado, sólo un supuesto es válido.

De manera semejante, una red puede tener detección de portadora (5a) o no (5b). Por lo general, las LANs tienen detección de portadora. Sin embargo, las redes inalámbricas no la pueden

utilizar de manera efectiva porque tal vez no todas las estaciones estén dentro del rango de radio de las demás. Las estaciones en redes alámbricas con detección de portadora pueden terminar su transmisión prematuramente si descubren que está chocando con otra transmisión. Por razones de ingeniería, la detección de colisión se hace muy rara vez en redes inalámbricas. Note que la palabra "portadora" en este sentido se refiere a una señal eléctrica en el cable y no tiene nada que ver con las empresas portadoras (por ejemplo, las compañías telefónicas), que datan de los tiempos del Pony Express.

4.2 PROTOCOLOS DE ACCESO MÚLTIPLE

Se conocen muchos algoritmos para asignar un canal de acceso múltiple. En las siguientes secciones estudiaremos una muestra representativa de los más interesantes y daremos algunos ejemplos de su uso.

4.2.1 ALOHA

En la década de 1970, Norman Abramson y sus colegas de la Universidad de Hawaii inventaron un método novedoso y elegante para resolver el problema de asignación de canal. Desde entonces, su trabajo ha sido extendido por muchos investigadores (Abramson, 1985). Aunque el trabajo de Abramson, llamado sistema ALOHA, usó la radiodifusión basada en tierra, la idea básica es aplicable a cualquier sistema en el que usuarios no coordinados compiten por el uso de un solo canal compartido.

Analizaremos dos versiones de ALOHA: puro y ranurado. Difieren en si se divide o no el tiempo en ranuras discretas en las que deben caber todas las tramas. El ALOHA puro no requiere sincronización global del tiempo; el ALOHA ranurado sí.

ALOHA puro

La idea básica de un sistema ALOHA es sencilla: permitir que los usuarios transmitan cuando tengan datos por enviar. Por supuesto, habrá colisiones y las tramas en colisión se dañarán. Sin embargo, debido a la propiedad de retroalimentación de la difusión, un emisor siempre puede saber si la trama fue destruida o no escuchando el canal, de la misma manera que los demás usuarios. Con una LAN, la retroalimentación es inmediata; vía satélite, hay un retardo de 270 mseg antes de que el emisor sepa si la transmisión tuvo éxito. Si por alguna razón no es posible escuchar mientras se transmite, se necesitan confirmaciones de recepción. Si la trama fue destruida, el emisor simplemente espera un tiempo aleatorio y la envía de nuevo. El tiempo de espera debe ser aleatorio o las mismas tramas chocarán una y otra vez, en sincronía. Los sistemas en los cuales varios usuarios comparten un canal común de modo tal que puede dar pie a conflictos se conocen como sistemas de **contención.**

En la figura 4-1 se presenta un esbozo de la generación de tramas en un sistema ALOHA. Hemos hecho que todas las tramas sean de la misma longitud porque la velocidad real de transporte (*throughput*) de los sistemas ALOHA se maximiza al tener tramas con un tamaño uniforme en lugar de tramas de longitud variable.

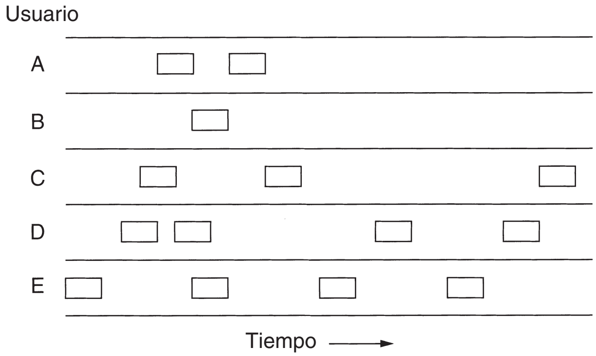


Figura 4-1. En ALOHA puro, las tramas se transmiten en momentos completamente arbitrarios.

Cada vez que dos tramas traten de ocupar el canal al mismo tiempo, habrá una colisión y ambas se dañarán. Si el primer bit de una trama nueva se traslapa con el último bit de una trama casi terminada, ambas tramas se destruirán por completo, y ambas tendrán que volver a transmitirse. La suma de verificación no puede (y no debe) distinguir entre una pérdida total y un error ligero. Lo malo es malo.

Una pregunta por demás interesante es: ¿cuál es la eficiencia de un canal ALOHA? Es decir, ¿qué fracción de todas las tramas transmitidas escapa a las colisiones en estas caóticas circunstancias? Primero consideremos un conjunto infinito de usuarios interactivos sentados ante sus computadoras (estaciones). Un usuario siempre está en uno de dos estados: escribiendo o esperando. Inicialmente, todos los usuarios están en el estado de escritura. Al terminar una línea, el usuario deja de escribir, en espera de una respuesta. La estación después transmite una trama que contiene la línea y verifica el canal para saber si llegó con éxito. De ser así, el usuario ve la respuesta y continúa escribiendo. Si no, el usuario continúa esperando y la trama se transmite una y otra vez hasta que se envía con éxito.

Denotemos con “tiempo de trama” el tiempo necesario para transmitir una trama estándar de longitud fija (es decir, la longitud de la trama dividida entre la tasa de bits). En este punto, suponemos que la población infinita de usuarios genera tramas nuevas según una distribución de Poisson con una media de N tramas por tiempo de trama. (La suposición de población infinita es necesaria para asegurar que N no disminuya a medida que se bloquean los usuarios.) Si $N > 1$, la comunidad de usuarios está generando tramas a una velocidad mayor que la que puede manejar el canal, y casi cada trama sufre una colisión. Para una velocidad real de transporte razonable esperaríamos que $0 < N < 1$.

Además de tramas nuevas, las estaciones también generan retransmisiones de tramas que previamente sufrieron colisiones. Asimismo, supongamos que la probabilidad de k intentos de

transmisión por tiempo de trama, viejas y nuevas combinadas, también es una distribución de Poisson, con una media de G por tiempo de trama. Claramente, $G \geq N$. Con carga baja (es decir, $N \approx 0$) habrá pocas colisiones y, por lo tanto, pocas retransmisiones, por lo que $G \approx N$. Con carga alta habrá muchas colisiones, por lo que $G > N$. Con todas las cargas, la velocidad real de transporte, S, es sólo la carga ofrecida, G, por la probabilidad, P_0, de que una transmisión tenga éxito (es decir, $S = GP_0$, donde P_0 es la probabilidad de que una trama no sufra una colisión).

Una trama no sufrirá una colisión si no se envían otras tramas durante un tiempo de trama desde su envío, como se muestra en la figura 4-2. ¿En qué condiciones llegará sin daño la trama sombreada? Sea t el tiempo requerido para enviar una trama. Si cualquier otro usuario generó una trama entre el tiempo t_0 y $t_0 + t$, el final de esa trama chocará con el comienzo de la trama sombreada. De hecho, el destino de la trama sombreada se selló aun antes de enviar el primer bit pero, dado que en ALOHA puro una estación no escucha el canal antes de transmitir, no tiene manera de saber que otra trama ya está en camino. De manera parecida, cualquier otra trama que salga entre $t_0 + t$ y $t_0 + 2t$ chocará con el final de la trama sombreada.

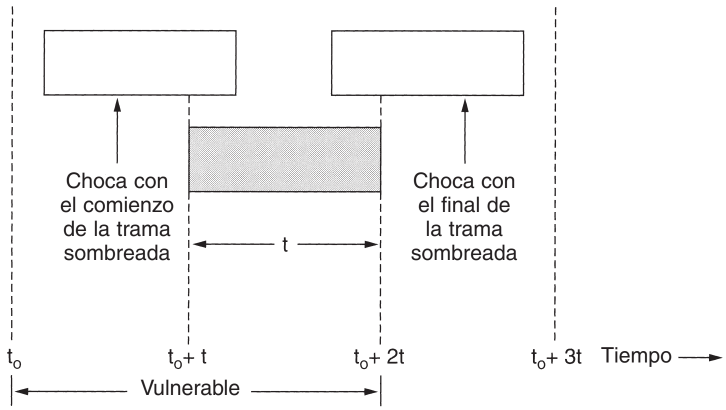


Figura 4-2. Periodo vulnerable para la trama sombreada.

La probabilidad de que k tramas sean generadas durante un tiempo de trama determinado está dada por la distribución de Poisson:

$$\Pr[k] = \frac{G^k e^{-G}}{k!} \tag{4-2}$$

así que la probabilidad de cero tramas es simplemente e^{-G}. En un intervalo de dos tiempos de trama de longitud, el número medio de tramas generadas es de $2G$. La probabilidad de que no se inicie otro tráfico durante todo el periodo vulnerable está dada entonces por $P_0 = e^{-2G}$. Si $S = GP_0$, obtenemos:

$$S = Ge^{-2G}$$

En la figura 4-3 se muestra la relación entre el tráfico ofrecido y la velocidad real de transporte. La velocidad real de transporte máxima ocurre a $G = 0.5$, con $S = 1/2e$, que es aproximadamen-

te 0.184. En otras palabras, lo más que podemos esperar es un uso del canal de 18%. Este resultado no es muy alentador, pero con todo mundo transmitiendo al azar difícilmente podríamos esperar una tasa de éxito de 100%.

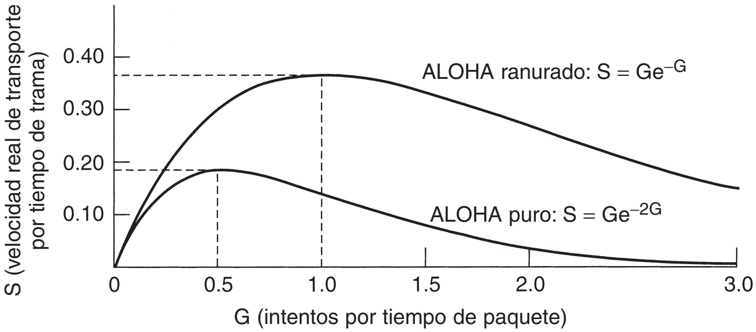


Figura 4-3. Velocidad real de transporte contra tráfico ofrecido en los sistemas ALOHA.

ALOHA ranurado

En 1972, Roberts publicó un método para duplicar la capacidad de un sistema ALOHA (Roberts, 1972). Su propuesta fue dividir el tiempo en intervalos discretos, cada uno de los cuales correspondía a una trama. Este enfoque requiere que los usuarios acuerden límites de ranura. Una manera de lograr la sincronización sería tener una estación especial que emitiera una señal al comienzo de cada intervalo, como un reloj.

En el método de Roberts, que se conoce como **ALOHA ranurado,** en contraste con el **ALOHA puro** de Abramson, no se permite que una computadora envíe cada vez que se pulsa un retorno de carro. En cambio, se le obliga a esperar el comienzo de la siguiente ranura. Por lo tanto, el ALOHA puro continuo se convierte en uno discreto. Dado que el periodo vulnerable ahora es de la mitad, la probabilidad de que no haya más tráfico durante la misma ranura que nuestra trama de prueba es de e^{-G}, lo que conduce a

$$S = Ge^{-G} \tag{4-3}$$

Como se puede ver en la figura 4-3, el ALOHA ranurado alcanza su máximo valor en $G = 1$, con una velocidad real de transporte de $S = 1/e$, o aproximadamente 0.368, el doble que el ALOHA puro. Si el sistema está operando a $G = 1$, la probabilidad de una ranura vacía es de 0.368 (de la ecuación 4-2). Lo mejor que podemos esperar usando ALOHA ranurado es 37% de ranuras vacías, 37% de éxitos y 26% de colisiones. La operación con valores mayores de G reduce el número de ranuras vacías pero aumenta de manera exponencial el número de colisiones. Para ver la manera en que se desarrolla este rápido crecimiento de colisiones con G, considere la transmisión de una trama de prueba. La probabilidad de que se evitará una colisión es de e^{-G}, que es la probabilidad de que los demás usuarios estén en silencio durante esa ranura. La probabilidad de una colisión es

entonces de sólo $1 - e^{-G}$. La probabilidad de que una transmisión requiera exactamente k intentos (es decir, $k - 1$ colisiones seguidas de un éxito) es

$$P_k = e^{-G}(1 - e^{-G})^{k-1}$$

El número esperado de transmisiones, E, por retorno de carro introducido es

$$E = \sum_{k=1}^{\infty} kP_k = \sum_{k=1}^{\infty} ke^{-G}(1 - e^{-G})^{k-1} = e^G$$

Como resultado de la dependencia exponencial de E respecto a G, pequeños aumentos en la carga del canal pueden reducir drásticamente su desempeño.

El Aloha ranurado es importante por una razón que al principio tal vez no sea obvia. Se diseñó en 1970 y se utilizó en algunos sistemas experimentales iniciales, después casi se olvidó por completo. Cuando se inventó el acceso a Internet a través de cable, de repente surgió el problema de cómo asignar un canal compartido entre varios usuarios competidores, por lo que el ALOHA ranurado se sacó del cesto de la basura para resolver el problema. Con frecuencia sucede que los protocolos que son perfectamente válidos caen en desuso por razones políticas (por ejemplo, alguna compañía grande desea que todos hagan las cosas a su manera), pero años más tarde alguna persona astuta se da cuenta de que un protocolo descartado por mucho tiempo es el que puede sacarlo de su problema actual. Por esta razón, en este capítulo estudiaremos varios protocolos elegantes que en la actualidad no se utilizan mucho, pero que podrían utilizarse fácilmente en aplicaciones futuras, siempre y cuando suficientes diseñadores de red los conozcan. Por supuesto, también estudiaremos varios protocolos que se utilizan en la actualidad.

4.2.2 Protocolos de acceso múltiple con detección de portadora

Con el ALOHA ranurado, el mejor aprovechamiento de canal que puede lograrse es $1/e$. Esto no es sorprendente pues, con estaciones que transmiten a voluntad propia, sin prestar atención a lo que están haciendo las demás estaciones, es inevitable que haya muchas colisiones. Sin embargo, en las redes de área local es posible que las estaciones detecten lo que están haciendo las demás estaciones y adapten su comportamiento con base en ello. Estas redes pueden lograr una utilización mucho mejor que $1/e$. En esta sección analizaremos algunos protocolos para mejorar el desempeño.

Los protocolos en los que las estaciones escuchan una portadora (es decir, una transmisión) y actúan de acuerdo con ello se llaman **protocolos de detección de portadora.** Se ha propuesto una buena cantidad de ellos. Kleinrock y Tobagi (1975) han analizado en forma detallada algunos de esos protocolos. A continuación mencionaremos varias versiones de los protocolos de detección de portadora.

CSMA persistente y no persistente

El primer protocolo de detección de portadora que estudiaremos aquí se llama **CSMA** (**Acceso Múltiple con Detección de Portadora**) **persistente-1.** Cuando una estación tiene datos por

transmitir, primero escucha el canal para saber si otra está transmitiendo en ese momento. Si el canal está ocupado, la estación espera hasta que se desocupa. Cuando la estación detecta un canal inactivo, transmite una trama. Si ocurre una colisión, la estación espera una cantidad aleatoria de tiempo y comienza de nuevo. El protocolo se llama persistente-1 porque la estación transmite con una probabilidad de 1 cuando encuentra que el canal está inactivo.

El retardo de propagación tiene un efecto importante en el desempeño del protocolo. Hay una pequeña posibilidad de que, justo después de que una estación comienza a transmitir, otra estación está lista para enviar y detectar el canal. Si la señal de la primera estación no ha llegado aún a la segunda, esta última detectará un canal inactivo y comenzará a enviar también, lo que dará como resultado una colisión. Cuanto mayor sea el tiempo de propagación, más importante será este efecto, y peor el desempeño del protocolo.

Aun si el retardo de propagación es cero, habrá colisiones. Si dos estaciones quedan listas a la mitad de la transmisión de una tercera, ambas esperarán respetuosamente hasta el fin de la transmisión y entonces comenzarán a transmitir de manera simultánea, lo que dará como resultado una colisión. Si no fueran tan impacientes, habría menos colisiones. Aun así, este protocolo es mucho mejor que el ALOHA puro, ya que ambas estaciones tienen la decencia de dejar de interferir con la trama de la tercera estación. Intuitivamente, esto conducirá a un mejor desempeño que el del ALOHA puro. Con el ALOHA ranurado ocurre exactamente lo mismo.

Un segundo protocolo de detección de portadora es el **CSMA no persistente.** En éste se hace un intento consciente por ser menos egoísta que en el previo. Antes de enviar, una estación escucha el canal. Si nadie más está transmitiendo, la estación comienza a hacerlo. Sin embargo, si el canal ya está en uso, la estación no lo escucha de manera continua a fin de tomarlo de inmediato al detectar el final de la transmisión previa. En cambio, espera un periodo aleatorio y repite el algoritmo. En consecuencia, este algoritmo conduce a un mejor uso del canal pero produce mayores retardos que el CSMA persistente-1.

El último protocolo es el **CSMA persistente-p,** que se aplica a canales ranurados y funciona como se explica a continuación. Cuando una estación está lista para enviar, escucha el canal. Si éste se encuentra inactivo, la estación transmite con una probabilidad p. Con una probabilidad $q = 1 - p$, se espera hasta la siguiente ranura. Si esa ranura también está inactiva, la estación transmite o espera nuevamente, con probabilidades p y q. Este proceso se repite hasta que la trama ha sido transmitida o hasta que otra estación ha comenzado a transmitir. En el segundo caso, la estación actúa como si hubiera habido una colisión (es decir, espera un tiempo aleatorio y comienza de nuevo). Si al inicio la estación detecta que el canal está ocupado, espera hasta la siguiente ranura y aplica el algoritmo anterior. En la figura 4-4 se muestra la velocidad real de transporte calculada contra el tráfico ofrecido para los tres protocolos, así como para el ALOHA puro y el ranurado.

CSMA con detección de colisiones

Los protocolos CSMA persistentes y no persistentes ciertamente son una mejora respecto a ALOHA porque aseguran que ninguna estación comienza a transmitir cuando detecta que el canal está ocupado. Otra mejora es que las estaciones aborten sus transmisiones tan pronto como detecten

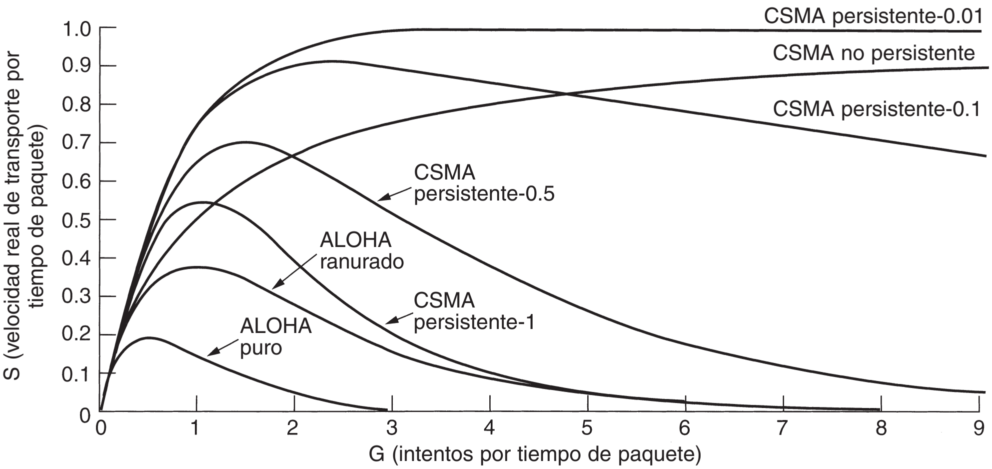


Figura 4-4. Comparación de la utilización del canal contra la carga para varios protocolos de acceso aleatorio.

una colisión. En otras palabras, si dos estaciones detectan que el canal está inactivo y comienzan a transmitir en forma simultánea, ambas detectarán la colisión casi de inmediato. En lugar de terminar de transmitir sus tramas, que de todos modos están irremediablemente alteradas, deben detener de manera abrupta la transmisión tan pronto como detectan la colisión. La terminación pronta de tramas dañadas ahorra tiempo y ancho de banda. Este protocolo, conocido como **CSMA/CD** (**Acceso Múltiple con Detección de Portadora y Detección de Colisiones),** se usa ampliamente en las LANs en la subcapa MAC. En particular, es la base de la popular LAN Ethernet, por lo que vale la pena dedicar un poco de tiempo a analizarlo con detalle.

CSMA/CD, al igual que muchos otros protocolos de LAN, utiliza el modelo conceptual de la figura 4-5. En el punto marcado t_0, una estación ha terminado de transmitir su trama. Cualquier otra estación que tenga una trama por enviar ahora puede intentar hacerlo. Si dos o más estaciones deciden transmitir en forma simultánea, habrá una colisión. Las colisiones pueden detectarse comparando la potencia o el ancho de pulso de la señal recibida con el de la señal transmitida.

Una vez que una estación detecta una colisión, aborta la transmisión, espera un tiempo aleatorio e intenta de nuevo, suponiendo que ninguna otra estación ha comenzado a transmitir durante ese lapso. Por lo tanto, nuestro modelo de CSMA/CD consistirá en periodos alternantes de contención y transmisión, ocurriendo periodos de inactividad cuando todas las estaciones están en reposo (por ejemplo, por falta de trabajo).

Ahora observemos con cuidado los detalles del algoritmo de contención. Suponga que dos estaciones comienzan a transmitir de manera exacta en el momento t_0. ¿En cuánto tiempo se darán cuenta de que ha habido una colisión? La respuesta a esta pregunta es vital para determinar la longitud del periodo de contención y, por lo tanto, el retardo y la velocidad real de transporte. El tiempo mínimo para detectar la colisión es sólo el tiempo que tarda la señal para propagarse de una estación a otra.

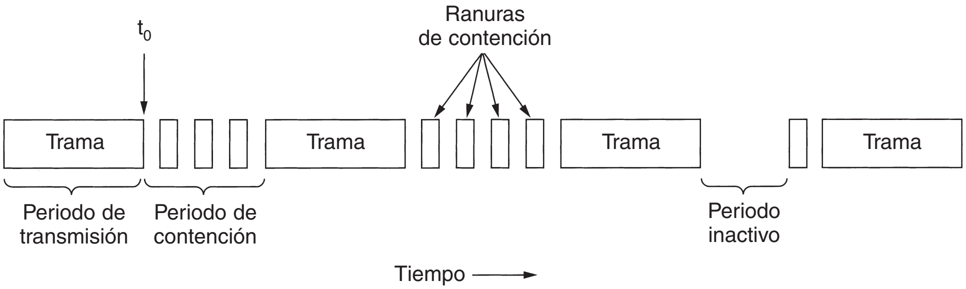


Figura 4-5. El CSMA/CD puede estar en uno de tres estados: contención, transmisión o inactivo.

Con base en este razonamiento, se podría pensar que una estación que después de iniciar su transmisión no detecta una colisión durante un tiempo igual al tiempo completo de propagación del cable podrá estar segura de que ha tomado el cable. Por "tomado" queremos decir que todas las demás estaciones sabían que estaba transmitiendo y no interfirieron. Esta conclusión es equivocada. Considere el siguiente escenario de peor caso. Sea τ el tiempo que tarda una señal en propagarse entre las dos estaciones más lejanas. En t_0, una estación comienza a transmitir. En $\tau - \varepsilon$, un instante antes de que la señal llegue a la estación más lejana, esa estación también comienza a transmitir. Por supuesto, detecta la colisión casi de inmediato y se detiene, pero la pequeña ráfaga de ruido causada por la colisión no regresa a la estación original hasta $2\tau - \varepsilon$. En otras palabras, en el peor caso una estación no puede estar segura de que ha tomado el canal hasta que ha transmitido durante 2τ sin detectar una colisión. Por esta razón, modelaremos el intervalo de contención como un sistema ALOHA ranurado con un ancho de ranura de 2τ. En un cable coaxial de 1 km de longitud, $\tau \approx 5$ µseg. Por sencillez supondremos que cada ranura contiene sólo 1 bit. Por supuesto, una vez que ha tomado el canal, una estación puede transmitir a cualquier tasa que desee, no sólo a 1 bit cada 2τ seg.

Es importante darse cuenta de que la detección de colisiones es un proceso *analógico*. El hardware de la estación debe escuchar el cable mientras transmite. Si lo que lee es distinto de lo que puso en él, sabe que está ocurriendo una colisión. La implicación es que la codificación de la señal debe permitir que se detecten colisiones (por ejemplo, una colisión de dos señales de 0 voltios bien podría ser imposible de detectar). Por esta razón, por lo general se usa una codificación especial.

También vale la pena mencionar que una estación emisora debe monitorear de manera continua el canal en busca de ráfagas de ruido que puedan indicar una colisión. Por esta razón, CSMA/CD con un solo canal es inherentemente un sistema semidúplex. Es imposible que una estación transmita y reciba tramas al mismo tiempo, debido a que la lógica de recepción está en uso, en busca de colisiones durante cada transmisión.

Para evitar cualquier malentendido, es importante hacer notar que ningún protocolo de subcapa MAC garantiza la entrega confiable. Incluso en ausencia de colisiones, el receptor podría no haber copiado en forma correcta la trama por varias razones (por ejemplo, falta de espacio de búfer o una interrupción no detectada).

4.2.3 Protocolos libres de colisiones

Aunque las colisiones no ocurren en CSMA/CD una vez que una estación ha tomado sin ambigüedades el canal, aún pueden ocurrir durante el periodo de contención. Estas colisiones afectan en forma adversa el desempeño del sistema, especialmente cuando el cable es largo (es decir, τ es grande) y las tramas cortas. Además, CSMA/CD no es aplicable en todo el mundo. En esta sección examinaremos algunos protocolos que resuelven la contención por el canal sin que haya colisiones, ni siquiera durante el periodo de contención. En la actualidad, la mayoría de ellos no se utilizan en los sistemas grandes, pero en un campo en constante cambio, el hecho de tener algunos protocolos con excelentes propiedades disponibles para sistemas futuros con frecuencia es algo bueno.

En los protocolos que describiremos suponemos que hay N estaciones, cada una con una dirección única de 0 a $N - 1$ incorporada en hardware. El hecho de que algunas estaciones puedan estar inactivas parte del tiempo no importa. También damos por hecho que el retardo de propagación no importa. La pregunta básica persiste: ¿qué estación toma el canal tras una transmisión exitosa? Continuamos usando el modelo de la figura 4-5 con sus intervalos de contención discretos.

Un protocolo de mapa de bits

En nuestro primer protocolo libre de colisiones, el **método básico de mapa de bits,** cada periodo de contención consiste en exactamente N ranuras. Si la estación 0 tiene una trama por enviar, transmite un bit 1 durante la ranura 0. No está permitido a ninguna otra estación transmitir durante esta ranura. Sin importar lo que haga la estación 0, la estación 1 tiene la oportunidad de transmitir un 1 durante la ranura 1, pero sólo si tiene en cola una trama. En general, la estación j puede anunciar que tiene una trama por enviar introduciendo un bit 1 en la ranura j. Una vez que han pasado las N ranuras, cada estación sabe cuáles son todas las estaciones que quieren transmitir. En ese punto, las estaciones comienzan a transmitir en orden numérico (vea la figura 4-6).



Figura 4-6. Protocolo básico de mapa de bits.

Dado que todos están de acuerdo en quién continúa, nunca habrá colisiones. Una vez que la última estación lista haya transmitido su trama, evento que pueden detectar fácilmente todas las estaciones, comienza otro periodo de contención de N bits. Si una estación queda lista justo después de que ha pasado su ranura de bits, ha tenido mala suerte y deberá permanecer inactiva hasta que cada estación haya tenido su oportunidad y el mapa de bits haya comenzado de nuevo. Los

protocolos como éste en los que el deseo de transmitir se difunde antes de la transmisión se llaman **protocolos de reservación.**

Analicemos con brevedad el desempeño de este protocolo. Por conveniencia, mediremos el tiempo en unidades de la ranura de bits de contención, con tramas de datos consistentes en d unidades de tiempo. En condiciones de carga baja, el mapa de bits simplemente se repetirá una y otra vez, debido a la falta de tramas de datos.

Considere la situación desde el punto de vista de una estación de número bajo, como 0 o 1. Por lo general, cuando la estación queda lista para transmitir, la ranura "actual" estará en algún lugar a la mitad del mapa de bits. En promedio, la estación tendrá que esperar $N/2$ ranuras para que el barrido actual termine, y otras N ranuras para que el siguiente barrido se ejecute hasta su terminación, antes de poder comenzar a transmitir.

Las perspectivas para las estaciones de número alto son mejores. En general, éstas sólo tendrán que esperar la mitad de un barrido ($N/2$ ranuras de bits) antes de comenzar a transmitir. Las estaciones de número alto pocas veces tienen que esperar el siguiente barrido. Dado que las estaciones de número bajo deben esperar un promedio de $1.5N$ ranuras y las estaciones de número alto deben esperar en promedio $0.5N$ ranuras, la media de todas las estaciones es de N ranuras. La eficiencia del canal cuando la carga es baja es fácil de calcular. La sobrecarga por trama es de N bits, y la cantidad de datos es de d bits, dando una eficiencia de $d/(N + d)$.

Si la carga es alta y todas las estaciones tienen algo que enviar todo el tiempo, el periodo de contención de N bits se prorratea entre N tramas, arrojando una sobrecarga de sólo 1 bit por trama, o una eficiencia de $d/(d + 1)$. El retardo medio de una trama es igual a la suma del tiempo que está en cola en su estación más un $N(d + 1)/2$ adicional una vez que llega a la cabeza de su cola interna.

Conteo descendente binario

Un problema con el protocolo básico de mapa de bits es que la sobrecarga es de 1 bit por estación, por lo que no se escala bien en redes con miles de estaciones. Podemos tener mejores resultados usando direcciones de estación binarias. Una estación que quiere utilizar el canal ahora difunde su dirección como una cadena binaria de bits, comenzando por el bit de orden mayor. Se supone que todas las direcciones tienen la misma longitud. A los bits en cada posición de dirección de las diferentes estaciones se les aplica un OR BOOLEANO a todos juntos. A este protocolo lo llamaremos **conteo descendente binario.** Se utilizó en Datakit (Fraser, 1987). Asume de manera implícita que los retardos de transmisión son insignificantes, de manera que todas las estaciones ven los bits instantáneamente.

Para evitar conflictos, debe aplicarse una regla de arbitraje: una vez que una estación ve que una posición de bit de orden alto, que en su dirección es 0, ha sido sobrescrita con un 1, se da por vencida. Por ejemplo, si las estaciones 0010, 0100, 1001 y 1010 están tratando de obtener el canal, en el primer tiempo de bit las estaciones transmiten 0, 0, 1 y 1, respectivamente. A éstos se les aplica el OR para formar un 1. Las estaciones 0010 y 0100 ven el 1 y saben que una estación de número mayor está compitiendo por el canal, por lo que se dan por vencidas durante esta ronda. Las estaciones 1001 y 1010 continúan.

El siguiente bit es 0, y ambas estaciones continúan. El siguiente bit es 1, por lo que la estación 1001 se da por vencida. La ganadora es la estación 1010, debido a que tiene la dirección mayor. Tras ganar la contienda, ahora puede transmitir una trama, después de lo cual comienza otro ciclo de contienda. El protocolo se ilustra en la figura 4-7. Tiene la propiedad de que estaciones con números grandes tienen una prioridad mayor que las estaciones con números pequeños, lo cual puede ser bueno o malo, dependiendo del contexto.

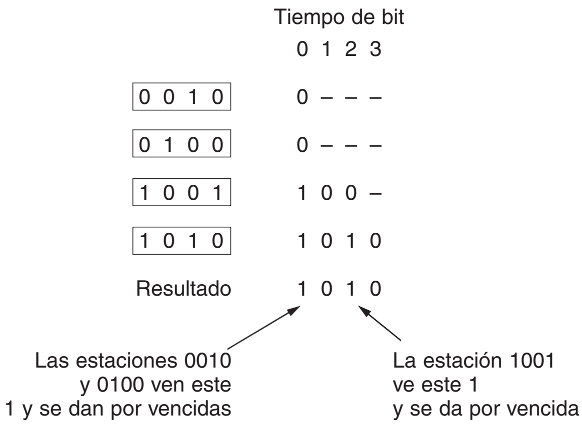


Figura 4-7. Protocolo de conteo descendente binario. Los guiones indican silencios.

La eficiencia de canal de este método es de $d/(d + \log_2 N)$. Sin embargo, si el formato de trama se escoge ingeniosamente de modo que la dirección del emisor sea el primer campo de la trama ni siquiera estos $\log_2 N$ bits se desperdician, y la eficiencia es de 100%.

Mok y Ward (1979) han descrito una variación del conteo descendente binario usando una interfaz paralela en lugar de serial. También sugieren el uso de números virtuales de estación, en el que los números virtuales de estación desde 0 hasta, e incluyendo, el de la estación ganadora se permutan en forma circular tras cada transmisión a fin de darle mayor prioridad a las estaciones que han estado en silencio demasiado tiempo. Por ejemplo, si las estaciones *C*, *H*, *D*, *A*, *G*, *B*, *E*, *F* tienen prioridades 7, 6, 5, 4, 3, 2, 1 y 0, respectivamente, entonces una transmisión exitosa de *D* la pone al final de la lista, dando un orden de prioridad de *C*, *H*, *A*, *G*, *B*, *E*, *F*, *D*. Por lo tanto, *C* permanece como la estación virtual 7, pero *A* sube de 4 a 5 y *D* desciende de 5 a 0. La estación *D* ahora sólo será capaz de adquirir el canal si ninguna otra estación lo quiere.

El conteo descendente binario es un ejemplo de un protocolo sencillo, elegante y eficiente que está esperando a ser redescubierto. Esperemos que algún día encuentre un nuevo hogar.

4.2.4 Protocolos de contención limitada

Hasta ahora hemos considerado dos estrategias básicas de adquisición del canal en una red de cable: los métodos por contención, como el CSMA, y los libres de colisión. Cada estrategia puede ser clasificada según lo bien que funciona en relación con las dos medidas de desempeño

importantes, el retardo con carga baja y la eficiencia del canal con carga alta. En condiciones de carga baja, la contención (es decir, ALOHA puro o ranurado) es preferible debido a su bajo retardo. A medida que aumenta la carga, la contención se vuelve paulatinamente menos atractiva, debido a que la sobrecarga asociada al arbitraje del canal se vuelve mayor. Lo inverso se cumple para los protocolos libres de colisiones. Con carga baja, tienen un retardo alto, pero a medida que aumenta la carga, mejora la eficiencia del canal en lugar de empeorar, como ocurre con los protocolos de contención.

Obviamente, sería agradable si pudiéramos combinar las mejores propiedades de los protocolos de contención y los libres de colisiones, desarrollando un protocolo nuevo que usara contención cuando la carga fuera baja para así tener un retardo bajo, y una técnica libre de colisiones cuando la carga fuera alta para lograr una buena eficiencia de canal. De hecho, existen tales protocolos, a los que llamaremos **protocolos de contención limitada,** y concluirán nuestro estudio de las redes de detección de portadora.

Hasta ahora, los únicos protocolos de contención que hemos estudiado han sido simétricos; es decir, cada estación intenta adquirir el canal con alguna probabilidad, p, y todas las estaciones usan la misma p. Resulta interesante que el desempeño general del sistema a veces puede mejorarse usando un protocolo que asigne diferentes probabilidades a diferentes estaciones.

Antes de ver los protocolos asimétricos, repasemos con rapidez el desempeño en el caso simétrico. Suponga que k estaciones están contendiendo por el acceso al canal. Cada una tiene una probabilidad p de transmitir durante cada ranura. La probabilidad de que una estación adquiera con éxito el canal durante una ranura dada es entonces de $kp(1-p)^{k-1}$. Para encontrar el valor óptimo de p, diferenciamos con respecto de p, igualamos el resultado a cero y despejamos p. Al hacerlo, encontramos que el mejor valor de p es $1/k$. Sustituyendo $p = 1/k$, obtenemos:

$$\text{Pr[éxito con } p \text{ óptima]} = \left[\frac{k-1}{k}\right]^{k-1} \qquad (4\text{-}4)$$

En la figura 4-8 se presenta gráficamente esta probabilidad. Para un número pequeño de estaciones, la probabilidad de éxito es buena, pero tan pronto la cantidad de estaciones alcanza cinco, la probabilidad cae a su valor asintótico de $1/e$.

En la figura 4-8 es bastante evidente que la probabilidad de que una estación adquiera el canal sólo puede aumentar disminuyendo la cantidad de competencia. Los protocolos de contención limitada hacen precisamente eso. Primero dividen las estaciones en grupos (no necesariamente separados). Sólo los miembros del grupo 0 pueden competir por la ranura 0. Si uno de ellos tiene éxito, adquiere el canal y transmite su trama. Si la ranura permanece desocupada o si hay una colisión, los miembros del grupo 1 compiten por la ranura 1, etcétera. Al dividir en forma adecuada y en grupos las estaciones, se puede reducir el grado de contención para cada ranura y, en consecuencia, se puede operar cada ranura cerca de la parte izquierda de la figura 4-8.

El truco está en cómo asignar las estaciones a las ranuras. Antes de ver el caso general, consideremos algunos casos especiales. En un extremo, cada grupo tiene sólo un miembro. Una asignación de este tipo garantiza que nunca habrá colisiones, pues cuando mucho una estación compite

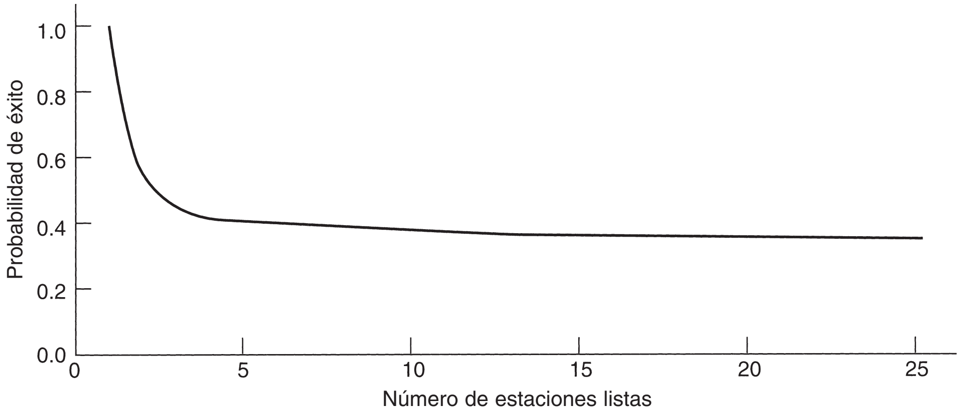


Figura 4-8. Probabilidad de adquisición de un canal de contención simétrica.

por una ranura dada. Antes vimos tales protocolos (por ejemplo, el conteo descendente binario). El siguiente caso especial es asignar dos estaciones por grupo. La probabilidad de que ambos intentarán transmitir durante una ranura es p^2, que para una p pequeña es insignificante. A medida que se asignan más estaciones a la misma ranura, aumenta la probabilidad de colisión, pero disminuye la longitud del barrido del mapa de bits necesaria para dar a todos una oportunidad. El caso límite es un solo grupo que contiene todas las estaciones (es decir, ALOHA ranurado). Lo que necesitamos es una manera de asignar de manera dinámica las estaciones a las ranuras, con muchas estaciones por ranura cuando la carga es baja y pocas estaciones (o incluso sólo una) por ranura cuando la carga es alta.

Protocolo de recorrido de árbol adaptable

Una manera particularmente sencilla de llevar a cabo la asignación necesaria es usar el algoritmo desarrollado por el ejército de Estados Unidos para hacer pruebas de sífilis a los soldados durante la Segunda Guerra Mundial (Dorfman, 1943). En pocas palabras, el ejército tomaba una muestra de sangre de N soldados. Se vaciaba una parte de cada muestra en un solo tubo de ensayo. Luego, esta muestra mezclada era examinada en busca de anticuerpos. Si no se encontraban, todos los soldados del grupo eran declarados sanos. Si se encontraban anticuerpos, se preparaban dos muestras mezcladas nuevas, una de los soldados 1 a $N/2$ y otra de los demás. El proceso se repetía recursivamente hasta que se determinaban los soldados infectados.

Para la versión de computadora de este algoritmo (Capetanakis, 1979) es conveniente considerar a las estaciones como hojas de un árbol binario, de la manera que se muestra en la figura 4-9. En la primera ranura de contención después de la transmisión satisfactoria de una trama, ranura 0, se permite que todas las estaciones intenten adquirir el canal. Si una de ellas lo logra, qué bueno. Si hay una colisión, entonces, durante la ranura 1, sólo aquellas estaciones que queden bajo el nodo 2 del árbol podrán competir. Si alguna de ellas adquiere el canal, la ranura que sigue a la trama se reserva para las estaciones que están bajo el nodo 3. Por otra parte, si dos o más

estaciones bajo el nodo 2 quieren transmitir, habrá una colisión durante la ranura 1, en cuyo caso es el turno del nodo 4 durante la ranura 2.

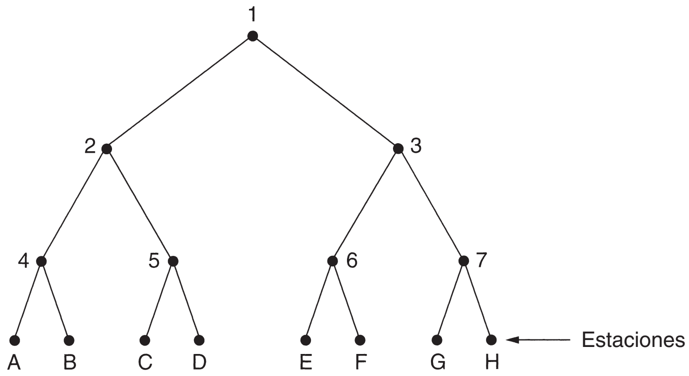


Figura 4-9. El árbol para ocho estaciones.

En esencia, si ocurre una colisión durante la ranura 0, se examina todo el árbol para localizar todas las estaciones listas. Cada ranura de bits está asociada a un nodo en particular del árbol. Si ocurre una colisión, continúa la búsqueda recursivamente con los hijos izquierdo y derecho del nodo. Si una ranura de bits está inactiva, o si sólo hay una estación que transmite en ella, puede detenerse la búsqueda de su nodo, pues se han localizado todas las estaciones listas. (Si hubiera existido más de una, habría ocurrido una colisión.)

Cuando la carga del sistema es pesada, apenas vale la pena dedicarle la ranura 0 al nodo 1, porque eso sólo tiene sentido en el caso poco probable de que precisamente una estación tenga una trama por enviar. De manera similar, se podría argumentar que los nodos 2 y 3 también podrían brincarse por la misma razón. En términos más generales, ¿en qué nivel del árbol debe comenzar la búsqueda? Es obvio que, a mayor carga, la búsqueda debe comenzar más abajo en el árbol. Supondremos que cada estación tiene una buena estimación del número de estaciones listas, q, por ejemplo, por la supervisión del tráfico reciente.

Para proceder, numeremos los niveles del árbol desde arriba, con el nodo 1 de la figura 4-9 en el nivel 0, los nodos 2 y 3 en el nivel 1, etcétera. Observe que cada nodo del nivel i tiene una fracción 2^{-i} de las estaciones por debajo de él. Si las q estaciones listas se distribuyen de manera uniforme, el número esperado de ellas por debajo de un nodo específico en el nivel i es sólo $2^{-i}q$. Intuitivamente, esperaríamos que el nivel óptimo para comenzar a examinar al árbol fuera aquel cuyo número medio de estaciones contendientes por ranura sea 1, es decir, el nivel en el que $2^{-i}q = 1$. Resolviendo esta ecuación, encontramos que $i = \log_2 q$.

Se han descubierto numerosas mejoras al algoritmo básico, y han sido analizadas con cierto detalle por Bertsekas y Gallager (1992). Por ejemplo, considere el caso en el que las estaciones G y H son las únicas que quieren transmitir. En el nodo 1 ocurrirá una colisión, por lo que se intentará el 2, pero se encontrará inactivo. No tiene caso probar el nodo 3, ya que está garantizado que tendrá una colisión (sabemos que dos o más estaciones bajo 1 están listas y que ninguna de ellas

está bajo 2, por lo que todas deben estar bajo 3). La prueba de 3 puede omitirse para intentar 6. Al no arrojar nada esta prueba, también puede omitirse 7 para intentar el nodo *G* después.

4.2.5 Protocolos de acceso múltiple por división de longitud de onda

Un método diferente para la asignación del canal es dividir el canal en subcanales usando FDM, TDM o ambas, y asignarlos de manera dinámica según se necesite. Por lo general, los esquemas como éste se usan en las LANs de fibra óptica para permitir que diferentes conversaciones usen distintas longitudes de onda (es decir, frecuencias) al mismo tiempo. En esta sección analizaremos uno de tales protocolos (Humblet y cols., 1992).

Una manera sencilla de construir una LAN completamente óptica es utilizar un acoplador pasivo en estrella (vea la figura 2-10). En efecto, se fusionan dos fibras de cada estación a un cilindro de vidrio. Una fibra es para salidas al cilindro y la otra para entradas del cilindro. La salida de luz de cualquier estación ilumina el cilindro y puede ser detectada por todas las demás estaciones. Las estrellas pasivas pueden manejar cientos de estaciones.

Para permitir múltiples transmisiones al mismo tiempo, se divide el espectro en canales (bandas de longitud de onda), como se muestra en la figura 2-31. En este protocolo, **WDMA** (**Acceso Múltiple por División de Longitud de Onda**), se asignan dos canales a cada estación. Se proporciona un canal estrecho como canal de control para señalizar la estación, y un canal ancho para que la estación pueda enviar tramas de datos.

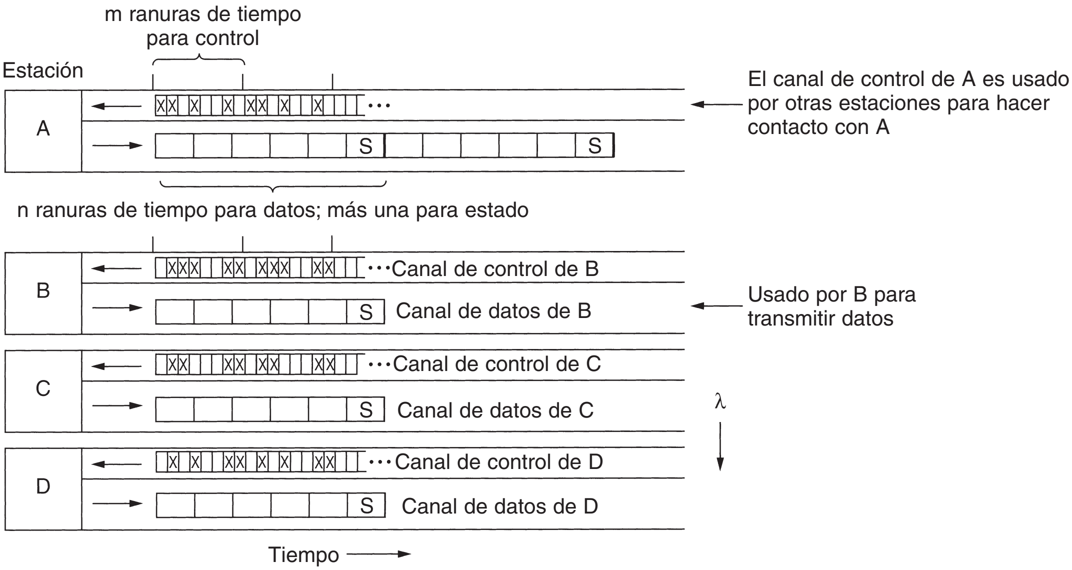


Figura 4-10. Acceso múltiple por división de longitud de onda.

Cada canal se divide en grupos de ranuras de tiempo, como se ilustra en la figura 4-10. Sea *m* el número de ranuras en el canal de control y $n + 1$ el número de ranuras en el canal de datos,

donde *n* de éstas son para datos y la última es usada por la estación para informar su estado (específicamente, qué ranuras de ambos canales están libres). En ambos canales, la secuencia de ranuras se repite de manera indefinida, marcándose la ranura 0 de una manera especial para que los que llegan tarde la puedan detectar. Todos los canales se sincronizan con un solo reloj global.

El protocolo reconoce tres clases de tráfico: (1) tráfico orientado a la conexión con tasa de datos constante, como vídeo sin comprimir, (2) tráfico orientado a la conexión con tasa de datos variable, como transferencia de archivos, y (3) tráfico de datagramas, como paquetes UDP. En los dos protocolos orientados a la conexión lo que se pretende es que para que *A* se comunique con *B*, primero debe introducir una trama de solicitud de conexión (CONNECTION REQUEST) en una ranura libre del canal de control de *B*. Si *B* lo acepta, la comunicación puede llevarse a cabo por el canal de datos de *A*.

Cada estación tiene dos emisores y dos receptores, como sigue:

1. Un receptor de longitud de onda fija para escuchar su propio canal de control.
2. Un emisor sintonizable para enviar por el canal de control de otra estación.
3. Un emisor de longitud de onda fija para la salida de tramas de datos.
4. Un receptor sintonizable para seleccionar al emisor de datos a escuchar.

En otras palabras, cada estación escucha en su propio canal de control las solicitudes que llegan, pero tiene que sintonizarse a la longitud de onda del emisor para obtener los datos. La sintonización de la longitud de onda se realiza con un interferómetro Fabry-Perot o Mach-Zehnder que filtra todas las longitudes de onda excepto la banda de longitud de onda deseada.

Ahora consideremos la manera en que la estación *A* establece un canal de comunicación clase 2 con la estación *B* para, digamos, transferencia de archivos. Primero, *A* sintoniza su receptor de datos con el canal de datos de *B* y espera la ranura de estado. Esta ranura indica cuáles ranuras de control están actualmente asignadas y cuáles están libres. Por ejemplo, en la figura 4-10 vemos que de las ocho ranuras de control de *B*, la 0, la 4 y la 5 están libres. Las demás están ocupadas (lo cual se indica por la "*x*").

A elige una de las ranuras de control libres, digamos la 4, e introduce ahí su mensaje de solicitud de conexión. Ya que *B* revisa de manera constante su canal de control, ve la solicitud y la acepta asignando la ranura 4 a *A*. Esta asignación se anuncia en la ranura de estado del canal de datos de *B*. Cuando *A* ve el anuncio, sabe que tiene una conexión unidireccional. Si *A* solicita una conexión bidireccional, *B* repite ahora el mismo algoritmo con *A*.

Es posible que, al mismo tiempo que *A* intente tomar la ranura de control 4 de *B*, *C* haga lo mismo. Ninguno lo conseguirá, y ambos se darán cuenta del fracaso revisando la ranura de estado del canal de control de *B*. Ahora ambos esperarán una cantidad de tiempo aleatoria y lo reintentarán después.

En este punto, cada parte tiene un mecanismo libre de conflictos para enviar mensajes de control cortos entre ellas. Para llevar a cabo la transferencia de archivos, *A* ahora envía a *B* un mensaje de control que dice, por ejemplo, "observa por favor mi siguiente salida de datos en la ranura 3. Hay

una trama de datos ahí para ti". Cuando *B* recibe el mensaje de control, sintoniza su receptor al canal de salida de *A* para leer la trama de datos. Dependiendo del protocolo de la capa superior, *B* puede utilizar el mismo mecanismo para regresar una confirmación de recepción, si lo desea.

Observe que surge un problema si tanto *A* como *C* tienen conexiones con *B* y cada uno de ellos le indica a *B* que busque en la ranura 3. *B* escogerá una de estas solicitudes al azar y la otra transmisión se perderá.

Para tráfico de tasa constante se utiliza una variación de este protocolo. Cuando *A* solicita una conexión, simultáneamente dice algo como: ¿Está bien si te envío una trama cada vez que ocurra la ranura 3? Si *B* puede aceptar (es decir, no tiene compromisos previos para la ranura 3), se establece una conexión de ancho de banda garantizado. Si no, *A* puede intentarlo después con una propuesta distinta, dependiendo de las ranuras de salida que tenga libres.

El tráfico clase 3 (datagramas) usa otra variación. En lugar de escribir un mensaje de solicitud de conexión en la ranura de control que acaba de encontrar (4), escribe un mensaje DATA FOR YOU IN SLOT 3 (HAY DATOS PARA TI EN LA RANURA 3). Si *B* está libre durante la siguiente ranura de datos 3, la transmisión tendrá éxito. De otra manera, se perderá la trama de datos. En esta forma, nunca se requieren conexiones.

Son posibles diversas variantes del protocolo. Por ejemplo, en lugar de dar a cada estación su propio canal de control, se puede compartir un solo canal de control entre todas las estaciones. Se asigna a cada estación un bloque de ranuras de cada grupo, multiplexando efectivamente varios canales virtuales en uno solo físico.

También es posible arreglárselas con un solo emisor y un solo receptor sintonizables por estación haciendo que el canal de cada estación se divida en m ranuras de control seguidas de $n + 1$ ranuras de datos. La desventaja aquí es que los emisores tienen que esperar más tiempo para capturar una ranura de control, y las tramas de datos consecutivas están más distantes porque se interpone cierta información de control.

Se han propuesto e implementado muchos otros protocolos de control WDMA, los cuales difieren en varios detalles. Algunos tienen sólo un canal de control, otros tienen varios. Algunos toman en cuenta el retardo de propagación, otros no. Algunos hacen del tiempo de sintonización una parte explícita del modelo, otros lo ignoran. Los protocolos también difieren en términos de complejidad de procesamiento, velocidad real de transporte y escalabilidad. Cuando se utiliza una gran cantidad de frecuencias, algunas veces este sistema se llama **DWDM** (**Multiplexión Densa por División de Longitud de Onda**). Para mayor información, vea (Bogineni y cols., 1993; Chen, 1994; Goralski, 2001; Kartalopoulos, 1999, y Levine y Akyildiz, 1995).

4.2.6 Protocolos de LANs inalámbricas

A medida que crece la cantidad de dispositivos de cómputo y comunicación portátiles, también crece la demanda de conexión de ellos con el mundo externo. Los primeros teléfonos portátiles ya tenían la capacidad de conectarse con otros teléfonos. Las primeras computadoras portátiles no tenían esta capacidad, pero poco después los módems se volvieron comunes en tales computadoras. Para entrar en línea, estas computadoras tenían que conectarse a un enchufe telefónico. El

requisito de una conexión con cable a la red fija significó que las computadoras eran portátiles, mas no móviles.

Para lograr una auténtica movilidad, las computadoras portátiles necesitan usar señales de radio (o infrarrojas) para comunicarse. De esta manera, los usuarios pueden leer y enviar correo electrónico mientras van de excursión o navegan. Un sistema de computadoras portátiles que se comunican por radio puede considerarse una LAN inalámbrica, como se trató en la sección 1.5.4. Estas LANs tienen propiedades un tanto diferentes que las LANs convencionales y requieren protocolos de subcapa MAC especiales. En esta sección estudiaremos algunos de estos protocolos. Puede encontrar más información sobre las LANs inalámbricas en (Geier, 2002, y O'Hara y Petrick, 1999).

Una configuración común para una LAN inalámbrica es un edificio de oficinas con estaciones base (también conocidas como puntos de acceso) ubicadas estratégicamente en distintas partes del edificio. Todas las estaciones base están interconectadas mediante cobre o fibra. Si la potencia de transmisión de las estaciones base y portátiles se ajusta a un alcance de 3 o 4 metros, entonces cada cuarto se vuelve una celda única, y el edificio entero se vuelve un sistema celular grande, como los sistemas de telefonía celular tradicionales que estudiamos en el capítulo 2. A diferencia de los sistemas telefónicos celulares, cada celda sólo tiene un canal, que cubre todo el ancho de banda disponible. Por lo general, el ancho de banda es de 11 a 54 Mbps.

En nuestros siguientes análisis haremos la suposición de simplificación de que todos los emisores de radio tienen algún alcance fijo. Cuando un receptor está dentro del alcance de dos emisores activos, la señal resultante por lo común se altera y resulta inútil, en otras palabras, ya no consideraremos los sistemas de tipo CDMA en este análisis. Es importante darse cuenta de que en algunas LANs inalámbricas no todas las estaciones están dentro del alcance de todas las demás, lo que conduce a una variedad de complicaciones. Es más, para las LANs inalámbricas interiores, la presencia de paredes entre las estaciones puede tener un impacto importante sobre el alcance efectivo de cada estación.

Un enfoque inocente para usar una LAN inalámbrica podría ser intentar el CSMA; escuchar si hay otras transmisiones y sólo transmitir si nadie más lo está haciendo. El problema radica en que este protocolo no es realmente adecuado porque lo que importa es la interferencia en el receptor, no en el emisor. Para ver la naturaleza de este problema, considere la figura 4-11, en la que se ilustran cuatro estaciones inalámbricas. Para nuestros fines, no importa cuáles son estaciones base ni cuáles son portátiles. El alcance de radio es tal que *A* y *B* están en el mismo alcance y potencialmente pueden interferir entre sí. *C* también podría interferir tanto con *B* como con *D*, pero no con *A*.



Figura 4-11. LAN inalámbrica. (a) *A* transmitiendo. (b) *B* transmitiendo.

Primero considere lo que ocurre cuando *A* está transmitiendo hacia *B*, como se muestra en la figura 4-11(a). Si *C* detecta el medio, no podrá escuchar a *A* porque está fuera de su alcance y, por tanto, deducirá falsamente que puede transmitir a *B*. Si *C* comienza a transmitir, interferirá en *B*, eliminando la trama de *A*. El problema de que una estación no puede detectar a un competidor potencial por el medio, puesto que dicho competidor está demasiado lejos, se denomina **problema de estación oculta.**

Ahora consideremos la situación inversa: *B* transmitiendo a *A*, como se muestra en la figura 4-11(b). Si *C* detecta el medio, escuchará una transmisión y concluirá equivocadamente que no puede enviar a *D*, cuando de hecho tal transmisión causaría una mala recepción sólo en la zona entre *B* y *C*, en la que no está localizado ninguno de los receptores pretendidos. Esta situación se denomina **problema de estación expuesta.**

El problema es que antes de comenzar una transmisión, una estación realmente necesita saber si hay actividad o no alrededor del receptor. El CSMA simplemente le indica si hay o no actividad alrededor de la estación que está detectando la portadora. Con un cable, todas las señales se propagan a todas las estaciones, de manera que sólo puede llevarse a cabo una transmisión en un momento dado en cualquier lugar del sistema. En un sistema basado en ondas de radio de corto alcance, pueden ocurrir transmisiones simultáneas si las ondas tienen destinos diferentes y éstos están fuera de alcance entre sí.

Otra forma de visualizar este problema es imaginar un edificio de oficinas en el que cada empleado tiene una computadora portátil inalámbrica. Suponga que Linda quiere enviar un mensaje a Melitón. La computadora de Linda detecta el entorno local y, al no percibir actividad, procede a transmitir. Sin embargo, aún puede hacer colisión en la oficina de Melitón, pues un tercero podría estar transmitiéndole actualmente desde una localidad tan alejada de la de Linda que la computadora de ésta no podría detectarlo.

MACA y MACAW

MACA (**Acceso Múltiple con Prevención de Colisiones**) (Karn, 1990) es uno de los primeros protocolos diseñados para LANs inalámbricas. El concepto en que se basa es que el emisor estimule al receptor a enviar una trama corta, de manera que las estaciones cercanas puedan detectar esta transmisión y eviten ellas mismas hacerlo durante la siguiente trama de datos (grande). El MACA se ilustra en la figura 4-12.

Consideremos ahora la manera en que *A* envía una trama a *B*. *A* comienza por enviar una trama **RTS** (**Solicitud de Envío**) a *B*, como se muestra en la figura 4-12(a). Esta trama corta (30 bytes) contiene la longitud de la trama de datos que seguirá posteriormente. Después *B* contesta con una trama **CTS** (**Libre para Envío**), como se muestra en la figura 4-12(b). La trama CTS contiene la longitud de los datos (copiada de la trama RTS). Una vez que sucede la recepción de la trama CTS, *A* comienza a transmitir.

Ahora veremos cómo reaccionan las estaciones que escuchan cualquiera de estas tramas. Cualquier estación que escuche el RTS evidentemente está bastante cerca de *A* y debe permanecer en silencio durante el tiempo suficiente para que el CTS se transmita de regreso a *A* sin conflicto. Cualquier estación que escuche el CTS está bastante cerca de *B* y debe permanecer en silencio

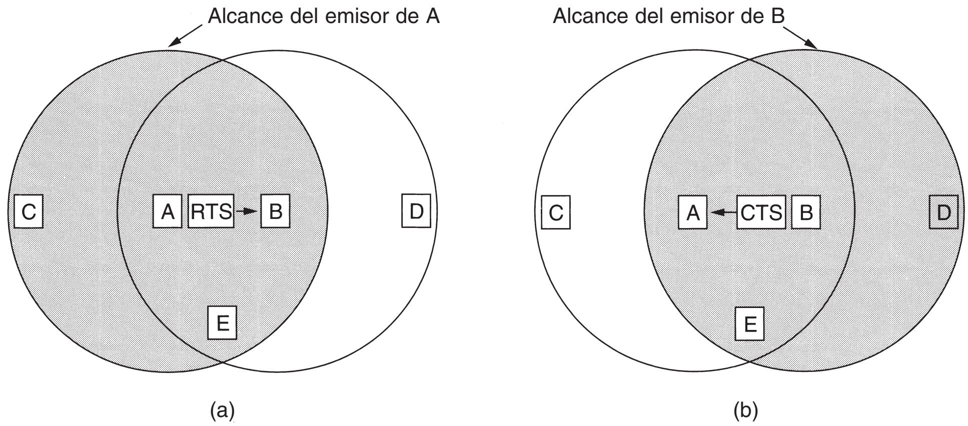


Figura 4-12. El protocolo MACA. (a) *A* enviando a *B* un RTS. (b) *B* respondiendo a *A* con un CTS.

durante la siguiente transmisión de datos, cuya longitud puede determinar examinando la trama CTS.

En la figura 4-12, *C* está en el alcance de *A*, pero no en el de *B*. Por lo tanto, escucha el RTS de *A* pero no el CTS de *B*. Mientras no interfiera con el CTS, está libre para transmitir mientras se está enviando la trama de datos. En contraste, *D* está en el alcance de *B* pero no de *A*. No escucha el RTS pero sí el CTS. Al escuchar el CTS se le indica que está cerca de una estación que está a punto de recibir una trama, por lo que difiere el envío de cualquier cosa hasta el momento en que se espera la terminación de esa trama. La estación *E* escucha ambos mensajes de control y, al igual que *D*, debe permanecer en silencio hasta que se haya completado la trama de datos.

A pesar de estas precauciones, aún pueden ocurrir colisiones. Por ejemplo, *B* y *C* pueden enviar tramas RTS a *A* al mismo tiempo. Éstas chocarán y se perderán. En el caso de una colisión, un emisor sin éxito (es decir, uno que no escucha un CTS en el intervalo de tiempo esperado) espera un tiempo aleatorio y reintenta. El algoritmo empleado es el retroceso exponencial binario, que estudiaremos cuando lleguemos a Ethernet.

Con base en estudios de simulación del MACA, Bharghavan y cols. (1994) afinaron el MACA para mejorar su desempeño y llamaron **MACAW** (**MACA Inalámbrico**) a su nuevo protocolo. Para comenzar, notaron que, sin confirmación de recepción de la capa de enlace de datos, las tramas no eran retransmitidas sino hasta que la capa de transporte notaba su ausencia, mucho después. Resolvieron este problema introduciendo una trama ACK tras cada trama de datos exitosa. También observaron que CSMA puede servir para evitar que una estación transmita un RTS al mismo tiempo y destino que otra estación cercana, por lo que se agregó la detección de portadora. Además, decidieron ejecutar el algoritmo de retroceso por separado para cada flujo de datos (par origen-destino), en lugar de para cada estación. Este cambio mejora la equidad del protocolo. Por último, agregaron un mecanismo para que las estaciones intercambiaran información sobre congestionamientos, y una manera de hacer que el algoritmo de retroceso reaccionara menos violentamente a problemas pasajeros, con lo que mejoraron el desempeño del sistema.

4.3 ETHERNET

Ya terminamos nuestra discusión general sobre los protocolos de asignación de canal, por lo que es tiempo de ver la forma en que estos principios se aplican a sistemas reales, particularmente a LANs. Como lo discutimos en la sección 1.5.3, el IEEE ha estandarizado varias redes de área local y de área metropolitana bajo el nombre de IEEE 802. Algunas han sobrevivido pero muchas no, como lo vimos en la figura 1-38. Quienes creen en la reencarnación piensan que Charles Darwin regresó como miembro de la Asociación de Estándares del IEEE para eliminar a los débiles. Los sobrevivientes más importantes son el 802.3 (Ethernet) y el 802.11 (LAN inalámbrica). Sería muy precipitado decir algo sobre el 802.15 (Bluetooth) y el 802.16 (MAN inalámbrica). Para mayor información, consulte la quinta edición de este libro. Tanto el 802.3 como el 802.11 tienen diferentes capas físicas y diferentes subcapas MAC, pero convergen en la misma subcapa de control lógico del enlace (que se define en el 802.2), por lo que tienen la misma interfaz a la capa de red.

En la sección 1.5.3 presentamos a Ethernet, por lo que no repetiremos aquí la información. En su lugar, nos enfocaremos en los detalles técnicos de Ethernet, en los protocolos y en los desarrollos recientes en la Ethernet de alta velocidad (gigabit). Puesto que Ethernet y el IEEE 802.3 son idénticos, excepto por dos diferencias mínimas que analizaremos pronto, muchas personas utilizan los términos "Ethernet" e "IEEE 802.3" de manera indistinta, y nosotros haremos lo mismo aquí.* Para información sobre Ethernet, vea (Breyer y Riley, 1999; Seifert, 1998, y Spurgeon, 2000).

4.3.1 Cableado Ethernet

Dado que el nombre "Ethernet" se refiere al cable (el éter), comencemos nuestro estudio por ahí. Comúnmente se usan cuatro tipos de cableado, como se muestra en la figura 4-13.

Nombre	Cable	Seg. máx.	Nodos/seg	Ventajas
10Base5	Coaxial grueso	500 m	100	Cable original; ahora obsoleto
10Base2	Coaxial delgado	185 m	30	No se necesita concentrador
10Base-T	Par trenzado	100 m	1024	Sistema más económico
10Base-F	Fibra óptica	2000 m	1024	Mejor entre edificios

Figura 4-13. Los tipos más comunes de cableado Ethernet.

Históricamente llegó primero el cable **10Base5,** llamado popularmente **Ethernet grueso;** semeja una manguera de jardín amarilla, con marcas cada 2.5 metros para indicar los puntos de las derivaciones. (El estándar 802.3 no *requiere* realmente que el cable sea amarillo, pero sí lo *sugiere*.) Por lo general, las conexiones a él se hacen usando **derivaciones vampiro,** en las que se introduce *cuidadosamente* una punta hasta la mitad del núcleo del cable coaxial. La notación

*Cabe mencionar que la comunicación entre estaciones que emplean Ethernet y estaciones que utilizan 802.3 no es posible aun cuando compartan el mismo canal (medio físico). (N. del R.T.)

10Base5 significa que opera a 10 Mbps, utiliza señalización de banda base y puede manejar segmentos de hasta 500 metros. El primer número es la velocidad en Mbps. Después viene la palabra "Base" (o algunas veces "BASE") para indicar transmisión de banda base. Solía haber una variante para banda ancha, 10Broad36, pero nunca tuvo popularidad en el mercado, por lo que desapareció. Por último, si el medio es coaxial, su longitud se da redondeada a unidades de 100 m después de "Base".

Históricamente, el segundo tipo de cable fue **10Base2** o **Ethernet delgado** que, a diferencia con el Ethernet grueso parecido a una manguera de jardín, se dobla con facilidad. Las conexiones se hacen usando conectores BNC estándar de la industria para formar uniones T, en lugar de emplear derivaciones vampiro. Los conectores son más fáciles de usar y más confiables. El Ethernet delgado es mucho más económico y fácil de instalar, pero sólo puede extenderse 185 metros por segmento, cada uno de los cuales puede manejar sólo 30 máquinas.

La detección de ruptura de cable, derivaciones malas o conectores flojos puede ser un problema importante en ambos medios. Por esta razón se han desarrollado técnicas para rastrear estos problemas. Básicamente, se inyecta un pulso de forma conocida en el cable. Si el pulso incide en un obstáculo o en el final del cable, se generará un eco que viajará de regreso. Si se cronometra cuidadosamente el intervalo entre el envío del pulso y la recepción del eco, es posible ubicar el origen del eco. Esta técnica se llama **reflectometría en el dominio del tiempo.**

Los problemas asociados con la localización de rupturas de cable han empujado a los sistemas a un tipo de patrón de cableado diferente, en el que todas las estaciones tienen cables que conducen a un **concentrador** (*hub*) central, en el que se conectan de manera eléctrica (como si se soldaran juntas). Por lo general, estos cables son pares trenzados telefónicos, ya que la mayoría de los edificios de oficinas ya están cableados de esta manera y normalmente hay bastantes pares extra disponibles. Este esquema se llama **10Base-T.** Los concentradores no almacenan en el búfer el tráfico de entrada. Más adelante en este capítulo analizaremos una versión mejorada de esta idea (conmutadores), la cual sí almacena en búfer el tráfico entrante.

Estos tres esquemas de cableado se ilustran en la figura 4-14. Para 10Base5, se sujeta firmemente un **transceptor** alrededor del cable, de modo que su derivación haga contacto con el núcleo interno. El transceptor contiene la electrónica que maneja detección de portadora y detección de colisiones. Al detectarse una colisión, el transceptor también pone una señal no válida especial en el cable para asegurar que todos los demás transceptores también se den cuenta de que ha ocurrido una colisión.

Con 10Base5, un **cable de transceptor** o **cable de derivación** conecta el transceptor a una tarjeta de interfaz en la computadora. El cable de transceptor puede tener hasta 50 metros de longitud y contiene cinco pares trenzados aislados individualmente. Dos de los pares son para entrada y salida de datos, respectivamente; dos más son para entrada y salida de señales de control. El quinto par, que no siempre se usa, permite que la computadora energice la electrónica del transceptor. Algunos transceptores permiten la conexión de hasta ocho computadoras cercanas a ellos, a fin de reducir la cantidad de transceptores necesarios.

El cable de transceptor termina en una tarjeta de interfaz en la computadora. La tarjeta de interfaz contiene un *chip* controlador que transmite tramas al transceptor y recibe tramas de él. El controlador se encarga de ensamblar los datos en el formato de trama adecuado, así como de

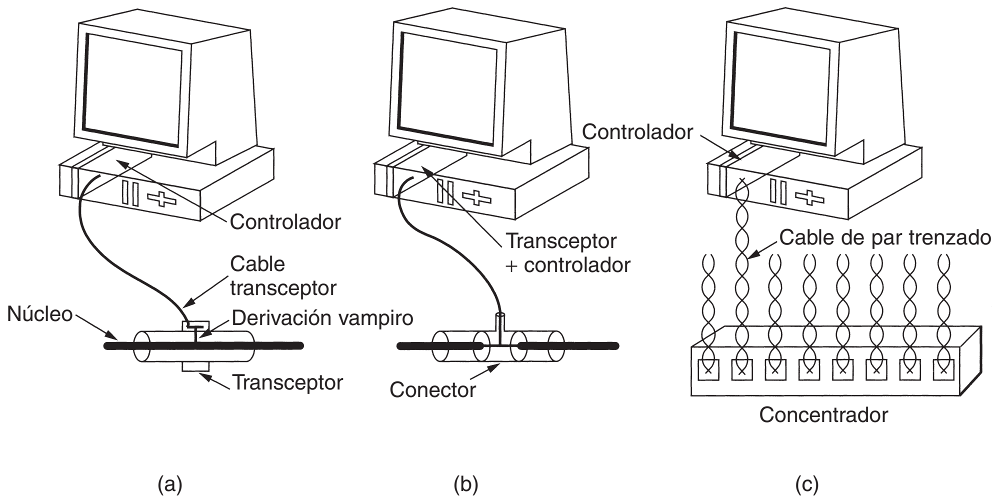


Figura 4-14. Tres tipos de cableado Ethernet. (a) 10Base5. (b) 10Base2. (c) 10Base-T.

calcular las sumas de verificación de las tramas de salida y de comprobarlas en las tramas de entrada. Algunos *chips* controladores también administran un grupo de búferes para las tramas de entrada, una cola para la transmisión de los búferes, transferencias de memoria directa con las computadoras *host* y otros aspectos de administración de la red.

Con 10Base2, la conexión al cable es sólo un conector BNC pasivo de unión T. La electrónica del transceptor está en la tarjeta controladora, y cada estación siempre tiene su propio transceptor.

Con 10Base-T no hay cable en absoluto, sólo el concentrador (una caja llena de circuitos electrónicos) al que cada estación está conectada mediante un cable dedicado (es decir, que no es compartido). Agregar o remover estaciones es más sencillo con esta configuración, y las rupturas de cable pueden detectarse con facilidad. La desventaja de 10Base-T es que la longitud máxima del cable es de sólo 100 metros, tal vez de 200 metros si se usa cable de par trenzado de alta calidad (categoría 5). Aun así, 10Base-T se está volviendo cada vez más común debido a que utiliza el cableado existente y a la facilidad de mantenimiento que ofrece. Se analizará una versión más rápida de 10Base-T (100Base-T) posteriormente en este capítulo.

Una cuarta opción de cableado para Ethernet es **10Base-F,** que usa fibra óptica. Esta alternativa es cara debido al costo de los conectores y los terminadores, pero tiene excelente inmunidad contra el ruido y es el método a usar para conexiones entre edificios o entre concentradores muy separados. Se permiten separaciones de kilómetros entre conexiones. También ofrece buena seguridad debido a que es más difícil intervenir una conexión de fibra que una de cobre.

En la figura 4-15 se muestran diferentes maneras de cablear un edificio. En la figura 4-15(a), un solo cable se pasa entre cuarto y cuarto, y cada estación se conecta a él en el punto más cercano. En la figura 4-15(b) una columna vertebral vertical corre del sótano a la azotea, y en cada piso se conectan cables horizontales a dicha columna mediante amplificadores especiales (repetidores). En algunos edificios los cables horizontales son delgados, y la red dorsal es gruesa. La topología

más general es en árbol, como en la figura 4-15(c), porque una red con dos rutas entre algunos pares de estaciones sufrirá interferencia entre las dos señales.

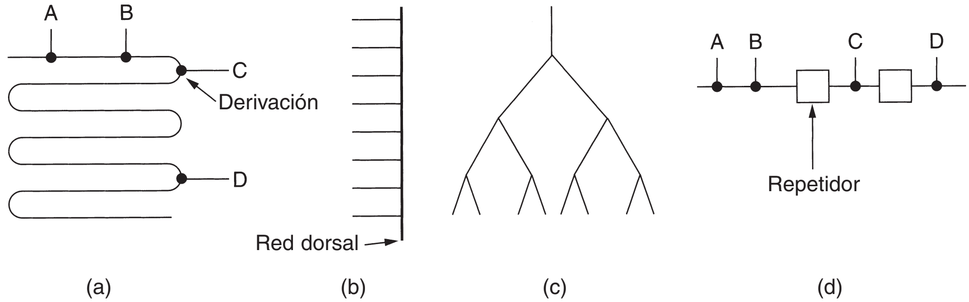


Figura 4-15. Topologías de cableado. (a) Lineal. (b) Columna vertebral. (c) Árbol. (d) Segmentada.

Cada versión de Ethernet tiene una longitud máxima de cable por segmento. Para permitir redes mayores, se pueden conectar múltiples cables mediante **repetidores,** como se muestra en la figura 4-15(d). Un repetidor es un dispositivo de capa física que recibe, amplifica (regenera) y retransmite señales en ambas direcciones. En lo que concierne al software, una serie de segmentos de cable conectados mediante repetidores no es diferente de un solo cable (excepto por el retardo introducido por los repetidores). Un sistema puede contener múltiples segmentos de cable y múltiples repetidores, pero ningún par de transceptores puede estar separado por más de 2.5 km y ninguna ruta entre dos transceptores puede atravesar más de cuatro repetidores.

4.3.2 Codificación Manchester

Ninguna de las versiones de Ethernet utiliza codificación binaria directa con 0 voltios para un bit 0 y 5 voltios para un bit 1, pues conduce a ambigüedades. Si una estación envía la cadena de bits 0001000, otros podrían interpretarla falsamente como 10000000 o 01000000, pues no pueden distinguir entre un emisor inactivo (0 voltios) y un bit 0 (0 voltios). Este problema se puede resolver utilizando +1 voltios para un 1 y −1 voltios para un 0, pero aún está el problema de que un receptor muestree la señal a una frecuencia ligeramente distinta a la que haya utilizado el emisor para generarla. Las diferentes velocidades de reloj pueden causar que el receptor y el emisor pierdan la sincronía respecto a dónde están los límites de bits, especialmente después de una serie larga de 0s consecutivos o una serie larga de 1s consecutivos.

Lo que se necesita es un mecanismo para que los receptores determinen sin ambigüedades el comienzo, el final o la mitad de cada bit sin referencia a un reloj externo. Dos de tales enfoques se llaman **codificación Manchester** y **codificación Manchester diferencial.** En la codificación Manchester, cada periodo de bit se divide en dos intervalos iguales. Un bit 1 binario se envía teniendo el voltaje alto durante el primer intervalo y bajo durante el segundo. Un 0 binario es justo lo inverso: primero bajo y después alto. Este esquema asegura que cada periodo de bit tenga una transición a la mitad, facilitando que el receptor se sincronice con el emisor. Una desventaja de la

codificación Manchester es que requiere el doble de ancho de banda que la codificación binaria directa, pues los pulsos son de la mitad de ancho. Por ejemplo, para enviar datos a 10 Mbps, la señal tiene que cambiar 20 millones de veces/seg. La codificación Manchester se muestra en la figura 4-16(b).

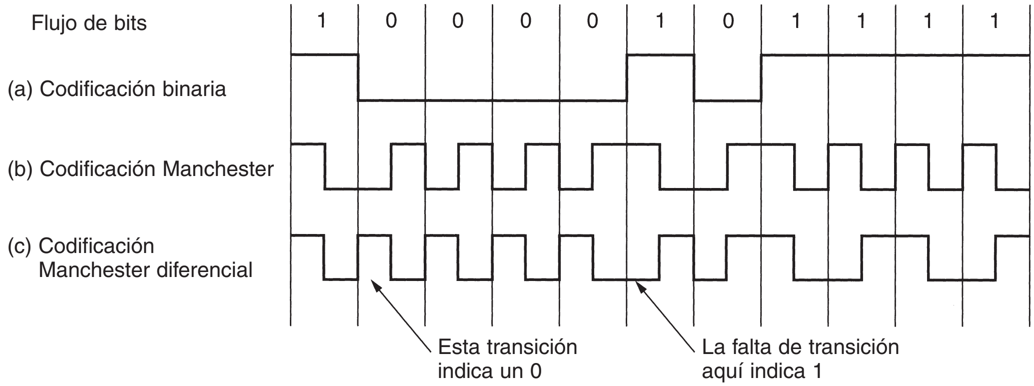


Figura 4-16. (a) Codificación binaria. (b) Codificación Manchester. (c) Codificación Manchester diferencial.

La codificación Manchester diferencial, que se muestra en la figura 4-16(c), es una variación de la codificación Manchester básica. En ella, un bit 1 se indica mediante la ausencia de una transición al inicio del intervalo. Un bit 0 se indica mediante la presencia de una transición al inicio del intervalo. En ambos casos también hay una transición a la mitad. El esquema diferencial requiere equipo más complejo, pero ofrece mejor inmunidad al ruido. Todos los sistemas Ethernet usan codificación Manchester debido a su sencillez. La señal alta es de + 0.85 voltios, y la señal baja es de − 0.85 voltios, dando un valor de DC de 0 voltios. Ethernet no utiliza la codificación Manchester diferencial, pero otras LANs (como la token ring 802.5) sí la utilizan.

4.3.3 El protocolo de subcapa MAC de Ethernet

En la figura 4-17 se muestra la estructura de trama original de DIX (DEC, Intel, Xerox). Cada trama inicia con un *Preámbulo* de 8 bytes, cada uno de los cuales contiene el patrón de bits 10101010. La codificación Manchester de este patrón produce una onda cuadrada de 10 MHz para 6.4 μseg para permitir que el reloj del receptor se sincronice con el del emisor. Se les pide que permanezcan sincronizados por el resto de la trama, utilizando la codificación Manchester para mantener un registro de los límites de bits.

La trama contiene dos direcciones, una para el destino y una para el origen. El estándar permite direcciones de 2 y 6 bytes, pero los parámetros definidos para el estándar de banda base de 10 Mbps usan sólo direcciones de 6 bytes. El bit de orden mayor de la dirección de destino es 0 para direcciones ordinarias y 1 para direcciones de grupo. Las direcciones de grupo permiten que

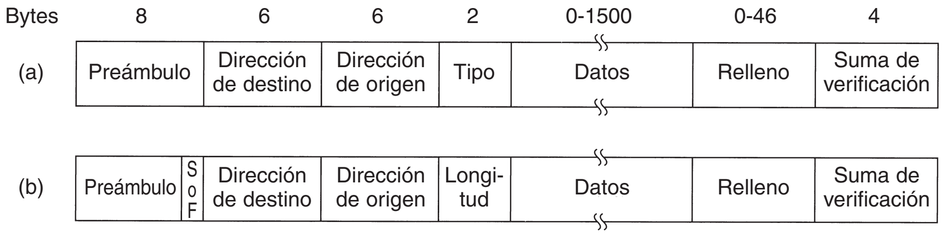


Figura 4-17. Formatos de trama. (a) Ethernet DIX. (b) IEEE 802.3.

varias estaciones escuchen en una sola dirección. Cuando una trama se envía a una dirección de grupo, todas las estaciones del grupo la reciben. El envío a un grupo de estaciones se llama **multidifusión** (*multicast*). La dirección que consiste únicamente en bits 1 está reservada para **difusión** (*broadcast*). Una trama que contiene sólo bits 1 en el campo de destino se acepta en todas las estaciones de la red. La diferencia entre difusión y multidifusión es lo suficientemente importante para garantizar la repetición. Una trama de multidifusión se envía a un grupo seleccionado de estaciones de la Ethernet; una trama de difusión se envía a todas las estaciones de la Ethernet. La multidifusión es más selectiva, pero involucra el manejo de grupos. La difusión es menos sofisticada pero no requiere manejo de grupos.

Otra característica interesante del direccionamiento es el empleo del bit 46 (adyacente al bit de orden mayor) para distinguir las direcciones locales de las globales. Las direcciones locales son asignadas por cada administrador de la red y no tienen significado fuera de la red local. En contraste, las direcciones globales son asignadas por el IEEE para asegurar que no haya dos estaciones en ningún lugar del mundo que tengan la misma dirección global. Con $48 - 2 = 46$ bits disponibles, hay unas 7×10^{13} direcciones globales. La idea es que cualquier estación pueda direccionar de manera exclusiva cualquier otra estación con sólo dar el número correcto de 48 bits. Es tarea de la capa de red encontrar la manera de localizar al destino.

A continuación está el campo de *Tipo*, que indica al receptor qué hacer con la trama. Es posible utilizar múltiples protocolos de capa de red al mismo tiempo en la misma máquina, por lo que cuando llega una trama de Ethernet, el *kernel* debe saber a cuál entregarle la trama. El campo de *Tipo* especifica a qué proceso darle la trama.

Después están los datos, de hasta 1500 bytes. Este límite fue elegido de manera algo arbitraria cuando se estableció el estándar DIX, principalmente con base en el hecho de que un transceptor necesita suficiente RAM para mantener toda una trama y la RAM era muy costosa en 1978. Un límite mayor podría haber significado más RAM y, por ende, un transceptor más costoso.

Además de haber una longitud de trama máxima, también hay una longitud mínima. Si bien algunas veces un campo de datos de 0 bytes es útil, causa problemas. Cuando un transceptor detecta una colisión, trunca la trama actual, lo que significa que los bits perdidos y las piezas de las tramas aparecen todo el tiempo en el cable. Para que Ethernet pueda distinguir con facilidad las tramas válidas de la basura, necesita que dichas tramas tengan una longitud de por lo menos 64 bytes, de la dirección de destino a la suma de verificación, incluyendo ambas. Si la porción de

datos de una trama es menor que 46 bytes, el campo de *Relleno* se utiliza para rellenar la trama al tamaño mínimo.

Otra razón (más importante) para tener una trama de longitud mínima es evitar que una estación complete la transmisión de una trama corta antes de que el primer bit llegue al extremo más alejado del cable, donde podría tener una colisión con otra trama. Este problema se ilustra en la figura 4-18. En el momento 0, la estación *A*, en un extremo de la red, envía una trama. Llamemos τ al tiempo que tarda en llegar esta trama al otro extremo. Justo antes de que la trama llegue al otro extremo (es decir, en el momento $\tau - \varepsilon$) la estación más distante, *B*, comienza a transmitir. Cuando *B* detecta que está recibiendo más potencia de la que está enviando, sabe que ha ocurrido una colisión, por lo que aborta su transmisión y genera una ráfaga de ruido de 48 bits para avisar a las demás estaciones. En otras palabras, atiborra el cable para asegurarse de que el emisor no ignore la colisión. En el momento 2τ, aproximadamente, el emisor ve la ráfaga de ruido y aborta también su transmisión; luego espera un tiempo aleatorio antes de reintentar.

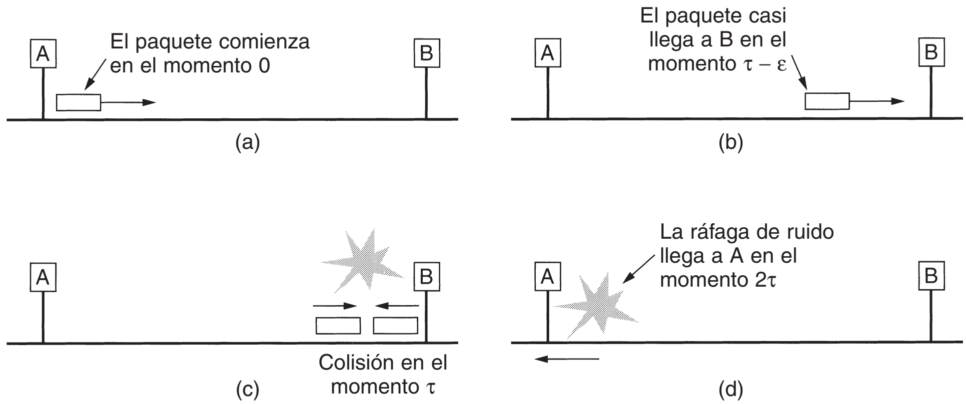


Figura 4-18. La detección de una colisión puede tardar hasta 2τ.

Si una estación intenta transmitir una trama muy corta, es concebible que ocurra una colisión, pero la transmisión se completa antes de que la ráfaga de ruido llegue de regreso, en el momento 2τ. El emisor entonces supondrá incorrectamente que la trama se envió con éxito. Para evitar que esta situación ocurra, todas las tramas deberán tardar más de 2τ para enviarse, de manera que la transmisión aún esté llevándose a cabo cuando la ráfaga de ruido regrese al emisor. Para una LAN de 10 Mbps con una longitud máxima de 2500 metros y cuatro repetidores (de la especificación 802.3), el tiempo de ida y vuelta (incluyendo el tiempo de propagación a través de los cuatro repetidores) se ha determinado a aproximadamente 50 μseg en el peor caso, incluyendo el tiempo para pasar a través de los repetidores, que con certeza es diferente de cero. Por lo tanto, la trama mínima debe tomar por lo menos este tiempo en transmitir. A 10 Mbps, un bit tarda 100 nseg, por lo que 500 bits es la trama más pequeña que se garantiza funcionará. Para agregar algún margen de seguridad, este número se redondeó a 512 bits o 64 bytes. Las tramas con menos de 64 bytes se rellenan con 64 bytes con el campo de *Relleno*.

A medida que aumente la velocidad de la red, la longitud mínima de la trama debe aumentar, o la longitud máxima del cable debe disminuir, de manera proporcional. Para una LAN de 2500 metros operando a 1 Gbps, el tamaño mínimo de trama tendría que ser de 6400 bytes. Como alternativa, el tamaño mínimo de trama podría ser de 640 bytes y la distancia máxima entre dos estaciones de 250 metros. Estas restricciones se vuelven cada vez más dolorosas a medida que progresamos hacia las redes de multigigabits.

El campo final de Ethernet es la *Suma de verificación*. De hecho, ésta es un código de *hash* de 32 bits de los datos. Si algunos bits de datos se reciben erróneamente (debido a ruido en el cable), es casi seguro que la suma de verificación está mal, y se detectará el error. El algoritmo de suma de verificación es una verificación de redundancia cíclica del tipo analizado en el capítulo 3. Simplemente realiza detección de errores, no corrección de errores hacia adelante.

Cuando el IEEE estandarizó Ethernet, el comité realizó dos cambios al formato DIX, como se muestra en la figura 4-17(b). El primero fue reducir el preámbulo a 7 bytes y utilizar el último byte para un delimitador de *Inicio de trama*, por compatibilidad con 802.4 y 802.5. El segundo fue cambiar el campo de *Tipo* en un campo de *Longitud*. Por supuesto, ahora no había forma de que el receptor supiera qué hacer con la trama entrante, pero ese problema se resolvió mediante la adición de un pequeño encabezado a la porción de datos para proporcionar esta información. Analizaremos el formato de la porción de datos cuando veamos el control lógico del enlace más adelante en este capítulo.

Desgraciadamente, en la época en que se publicó el 802.3 ya se utilizaba tanto hardware y software para la Ethernet DIX que muy pocos fabricantes y usuarios tenían deseos de convertir el campo de *Tipo* en uno de *Longitud*. En 1997, el IEEE tiró la toalla y dijo que las dos formas se ajustaban bien. Por fortuna, todos los campos de *Tipo* en uso antes de 1997 eran más grandes que 1500. En consecuencia, cualquier número menor que o igual a 1500 puede interpretarse como *Longitud*, y cualquier número más grande que 1500 puede interpretarse como *Tipo*. Ahora el IEEE puede afirmar que todo el mundo está utilizando su estándar y que cada quién puede seguir haciendo lo que desee sin preocuparse.

4.3.4 Algoritmo de retroceso exponencial binario

Ahora veamos cómo se efectúa el proceso de aleatorización cuando ocurre una colisión. El modelo es el de la figura 4-5. Tras una colisión, el tiempo se divide en ranuras discretas cuya longitud es igual al tiempo de propagación de ida y vuelta de peor caso en el cable (2τ). Tomando en cuenta la ruta más larga permitida por Ethernet, el tiempo de ranura se estableció en 512 tiempos de bit, o 51.2 μseg.

Tras la primera colisión, cada estación espera 0 o 1 tiempos de ranura antes de intentarlo de nuevo. Si dos estaciones entran en colisión y ambas escogen el mismo número aleatorio, habrá una nueva colisión. Después de la segunda colisión, cada una escoge 0, 1, 2 o 3 al azar y espera ese número de tiempos de ranura. Si ocurre una tercera colisión (la probabilidad de que esto ocurra es

de 0.25), entonces para la siguiente vez el número de ranuras a esperar se escogerá al azar del intervalo 0 a $2^3 - 1$.

En general, tras i colisiones, se escoge un número aleatorio entre 0 y $2^i - 1$, y se salta ese número de ranuras. Sin embargo, tras haberse alcanzado 10 colisiones, el intervalo de aleatorización se congela en un máximo de 1023 ranuras. Tras 16 colisiones, el controlador tira la toalla e informa de un fracaso a la computadora. La recuperación posterior es responsabilidad de las capas superiores.

Este algoritmo, llamado **retroceso exponencial binario,** se escogió para adaptar en forma dinámica el número de estaciones que intentan transmitir. Si el intervalo de aleatorización para todas las colisiones fuera de 1023, la posibilidad de que chocaran dos estaciones una segunda vez será insignificante, pero la espera promedio tras una colisión será de cientos de tiempos de ranura, lo que introduce un retardo significativo. Por otra parte, si cada estación siempre se retrasa 0 o 1 ranuras, entonces, al tratar de transmitir 100 estaciones al mismo tiempo, habría colisiones una y otra vez, hasta que 99 de ellas escogieran 1 y la estación restante escogiera 0. Esto podría tomar años. Haciendo que el intervalo de aleatorización crezca de manera exponencial a medida que ocurren más y más colisiones, el algoritmo asegura un retardo pequeño cuando sólo unas cuantas estaciones entran en colisión, pero también asegura que la colisión se resuelva en un intervalo razonable cuando hay colisiones entre muchas estaciones. Truncar el retroceso a 1023 evita que el límite crezca demasiado.

Como se ha descrito hasta ahora, CSMA/CD no proporciona confirmación de recepción. Ya que la simple ausencia de colisiones no garantiza que los bits no fueron alterados por picos de ruido en el cable, para una comunicación confiable el destino debe verificar la suma de verificación y, de ser correcta, regresar al origen una trama de confirmación de recepción. Por lo general, esta confirmación sería simplemente otra trama, en lo que concierne al protocolo, y tendría que pelear por tiempo de canal de la misma manera que una trama de datos. Sin embargo, una modificación sencilla del algoritmo de contención permite la confirmación rápida de la recepción de una trama (Tokoro y Tamaru, 1977). Todo lo que se necesitará es reservar para la estación de destino la primera ranura de contención que siga a la siguiente transmisión exitosa. Desgraciadamente, el estándar no proporciona esta posibilidad.

4.3.5 Desempeño de Ethernet

Ahora examinaremos brevemente el desempeño de Ethernet en condiciones de carga pesada y constante, es decir, k estaciones siempre listas para transmitir. Es complicado un análisis riguroso del algoritmo de retroceso exponencial binario. En cambio, seguiremos a Metcalfe y Boggs (1976) y supondremos una probabilidad constante de retransmisión en cada ranura. Si cada estación transmite durante una ranura de contención con una probabilidad p, la probabilidad A de que una estación adquiera el canal durante esa ranura es de:

$$A = kp(1 - p)^{k-1} \tag{4-5}$$

A se maximiza cuando $p = 1/k$, con $A \rightarrow 1/e$ conforme $k \rightarrow \infty$. La probabilidad de que el intervalo de contención tenga exactamente j ranuras es de $A(1 - A)^{j-1}$, por lo que el número medio de ranuras por contención está dado por

$$\sum_{j=0}^{\infty} jA(1 - A)^{j-1} = \frac{1}{A}$$

Puesto que cada ranura tiene una duración de 2τ, el intervalo medio de contención, w, es $2\tau/A$. Suponiendo una p óptima, el número medio de ranuras de contención nunca es mayor que e, por lo que w es, cuando mucho, $2\tau e \approx 5.4\tau$.

Si la trama media tarda P segundos en transmitirse, cuando muchas estaciones tienen tramas por enviar,

$$\text{Eficiencia del canal} = \frac{P}{P + 2\tau/A} \tag{4.6}$$

Aquí vemos dónde entra la distancia máxima de cable entre dos estaciones en las cifras de desempeño, dando lugar a topologías distintas de las de la figura 4-15(a). Cuanto mayor sea la longitud del cable, mayor será el intervalo de contención. Debido a esta observación el estándar Ethernet especifica una longitud máxima de cable.

Es instructivo formular la ecuación 4-6 en términos de la longitud de trama, F, el ancho de banda de la red, B, la longitud del cable, L, y la velocidad de propagación de la señal, c, para el caso óptimo de e ranuras de contención por trama. Con $P = F/B$, la ecuación 4-6 se convierte en

$$\text{Eficiencia del canal} = \frac{1}{1 + 2BLe/cF} \tag{4-7}$$

Cuando el segundo término del denominador es grande, la eficiencia de la red es baja. Más específicamente, un aumento en el ancho de banda o la distancia de la red (el producto BL) reduce la eficiencia de una trama de tamaño dado. Desgraciadamente, mucha investigación sobre hardware de redes está enfocada precisamente a aumentar este producto. La gente quiere un gran ancho de banda a través de distancias grandes (por ejemplo, en las MANs de fibra óptica), lo que sugiere que Ethernet tal vez no sea el mejor sistema para estas aplicaciones. Veremos otras formas de implementación de Ethernet cuando analicemos la Ethernet conmutada, más adelante en este capítulo.

En la figura 4-19 se presenta gráficamente la eficiencia del canal contra el número de estaciones listas para $2\tau = 51.2$ μseg y una tasa de datos de 10 Mbps usando la ecuación 4-7. Con un tiempo de ranura de 64 bytes, no es sorprendente que las tramas de 64 bytes no sean eficientes. Por otra parte, con tramas de 1024 bytes y un valor asintótico de e ranuras de 64 bytes por intervalo de contención, el periodo de contención tiene 174 bytes de longitud y la eficiencia es de 0.85.

Para determinar el número medio de estaciones listas para transmitir en condiciones de carga alta, podemos usar la siguiente (y burda) observación. Cada trama atrapa el canal durante un periodo de contención más un tiempo de transmisión de trama, lo que da un total de $P + w$ seg. El número de tramas por segundo, por lo tanto, es $1/(P + w)$. Si cada estación genera tramas a una

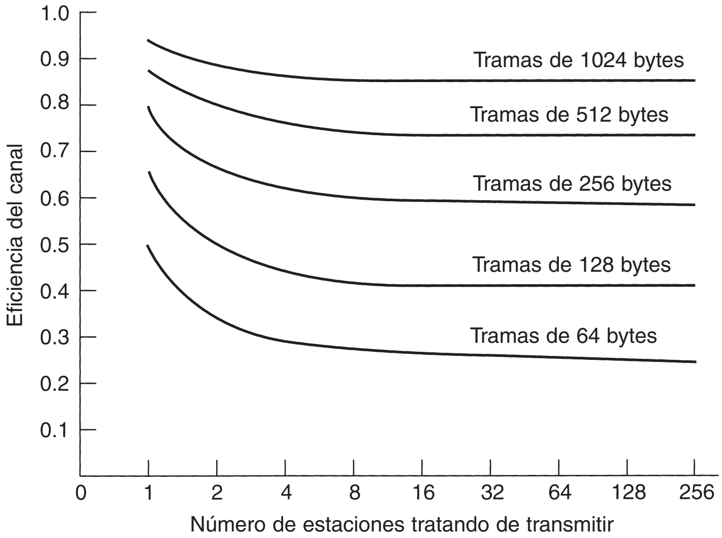


Figura 4-19. Eficiencia de Ethernet a 10 Mbps con tiempos de ranura de 512 bits.

velocidad media de λ tramas/seg, cuando el sistema está en el estado *k*, la tasa de entrada total de todas las estaciones no bloqueadas combinadas es de *k*λ tramas/seg. Ya que en equilibrio las tasas de entrada y de salida deben ser idénticas, podemos igualar esas dos expresiones y despejar *k*. (Observe que *w* es una función de *k*.) En (Bertsekas y Gallager, 1992) se da un análisis más elaborado.

Probablemente vale la pena mencionar que se ha realizado una gran cantidad de análisis teóricos de desempeño de Ethernet (y otras redes). Prácticamente todos estos trabajos han supuesto que el tráfico es Poisson. A medida que los investigadores han comenzado a examinar datos reales, se ha hecho evidente que el tráfico en redes pocas veces es Poisson, sino autosimilar (Paxson y Floyd, 1994, y Willinger y cols., 1995). Lo que esto significa es que el promedio durante periodos grandes no hace más uniforme el tráfico. La cantidad media de paquetes en cada minuto de una hora tiene tanta variación como la cantidad media de paquetes en cada segundo de un minuto. La consecuencia de este descubrimiento es que la mayoría de los modelos de tráfico de red no se aplican al mundo real y deben tomarse con escepticismo.

4.3.6 Ethernet conmutada

A medida que se agregan más y más estaciones a una Ethernet, aumenta el tráfico. En algún momento, la LAN se saturará. Una solución al problema es utilizar una velocidad mayor, digamos 100 Mbps en lugar de 10 Mbps. Pero con el crecimiento de la multimedia, incluso una Ethernet de 100 Mbps o de 1 Gbps puede saturarse.

Afortunadamente existe una solución diferente para tratar con el aumento de carga: una Ethernet conmutada, como la que se muestra en la figura 4-20. El corazón de este sistema es un **conmutador** (*switch*) que contiene una matriz de conmutación de alta velocidad y espacio

(típicamente) para 4 a 32 tarjetas de línea, cada una de las cuales contiene de uno a ocho conectores. Lo más común es que cada conector tenga una conexión de cable de par trenzado 10Base-T a una sola computadora *host*.

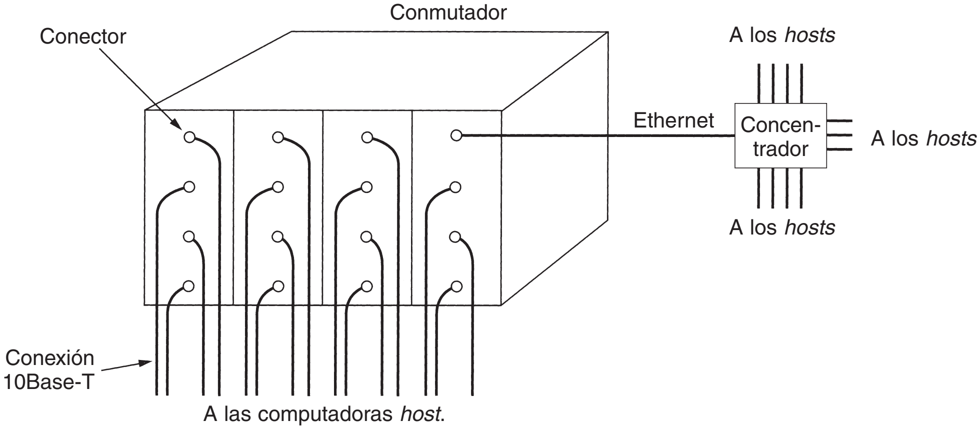


Figura 4-20. Ejemplo sencillo de Ethernet conmutada.

Cuando una estación quiere transmitir una trama Ethernet, envía una trama estándar al conmutador. La tarjeta que recibe la trama la revisa para ver si está destinada a una de las otras estaciones conectadas a la misma tarjeta. De ser así, la trama se copia ahí. Si no, la trama se envía a través de la matriz de conmutación de alta velocidad a la tarjeta de la estación de destino. Por lo general, dicha matriz de conmutación funciona a más de 1 Gbps usando un protocolo patentado.

¿Qué ocurre si dos máquinas conectadas a la misma tarjeta de conexión transmiten tramas al mismo tiempo? Depende de la manera en que haya sido construida la tarjeta. Una posibilidad es que todos los puertos de la tarjeta estén alambrados entre sí para formar una LAN local dentro de la tarjeta. Las colisiones en esta LAN en tarjeta se detectan y manejan igual que cualquier otra colisión en una red CSMA/CD, en las que las retransmisiones utilizan el algoritmo de retroceso exponencial binario. Con este tipo de tarjeta sólo es posible una transmisión por tarjeta en un momento dado, pero todas las tarjetas pueden estar transmitiendo en paralelo. Con este diseño, cada tarjeta forma su propio **dominio de colisión,** independiente de los demás. Con sólo una estación por dominio de colisión, las colisiones son imposibles y el desempeño se mejora.

Con el otro tipo de tarjeta de conexión, cada puerto de entrada se almacena en un búfer, por lo que las tramas de entrada se almacenan en la RAM de la tarjeta conforme llegan. Este diseño permite que todos los puertos de entrada reciban (y transmitan) tramas al mismo tiempo, para una operación en paralelo completamente dúplex, algo que no es posible con CSMA/CD en un solo canal. Una vez que se ha recibido por completo la trama, la tarjeta puede determinar si la trama está destinada a otro puerto de la misma tarjeta o a un puerto distante. En el primer caso, puede transmitirse directamente al destino. En el segundo, debe transmitirse a través de la matriz de conmutación de alta velocidad a la tarjeta apropiada. Con este diseño, cada puerto es un dominio de colisión independiente, por lo que no ocurren colisiones. En muchos casos, la velocidad real

de transporte total del sistema puede aumentarse en un orden de magnitud respecto al 10Base-5, que tiene un solo dominio de colisión para todo el sistema.

Dado que el conmutador sólo espera tramas Ethernet estándar en cada puerto de entrada, es posible utilizar algunos de los puertos como concentradores. En la figura 4-20, el puerto de la esquina superior derecha no está conectado a una sola estación, sino a un concentrador de 12 puertos. A medida que llegan las tramas al concentrador, luchan por el canal de la manera usual, con colisiones y retroceso binario. Las tramas que tienen éxito llegan al conmutador y ahí se tratan como cualquier otra trama de entrada: se conmutan a la línea de salida correcta a través de la matriz de conmutación de alta velocidad. Los concentradores son más baratos que los conmutadores, pero debido a los precios a la baja de los conmutadores, se están haciendo obsoletos rápidamente. No obstante, los concentradores heredados aún existen.

4.3.7 Fast Ethernet

Al principio, 10 Mbps parecían el cielo, al igual que los módems de 1200 bps parecieron el cielo a los primeros usuarios de módems acústicos de 300 bps. Pero la novedad desapareció rápidamente. Como un tipo de corolario a la Ley de Parkinson ("El trabajo se expande hasta agotar el tiempo destinado a realizarlo"), tal pareciera que los datos se expandieron hasta agotar el ancho de banda disponible para su transmisión. Para aumentar la velocidad, varios grupos de la industria propusieron dos nuevas LANs ópticas basadas en anillos. Una se llamó **FDDI** (**Interfaz de Datos Distribuidos por Fibra**) y la otra se llamó **Canal de fibra.** Para acortar la historia, si bien ambas se utilizaban como redes dorsales, ninguna se popularizó en sistemas de escritorio. En ambos casos, la administración de estaciones era muy complicada, lo que llevó a *chips* complejos y precios altos. La lección que debió aprenderse a partir de aquí fue KISS (*Keep It Simple, Stupid*; Mantenlo Simple, Tonto).

En cualquier caso, al no popularizarse las LANs ópticas quedó un hueco para redes Ethernet de una gran variedad a velocidades superiores a 10 Mbps. Muchas instalaciones necesitaban más ancho de banda y, por lo tanto, tenían numerosas LANs de 10 Mbps conectadas por una maraña de repetidores, puentes, enrutadores y puertas de enlace, aunque los administradores de redes algunas veces sentían que las conexiones parecían estar hechas con goma de mascar y tela metálica, es decir, endebles y poco seguras.

Fue en este entorno que el IEEE convocó al comité 802.3 en 1992 con instrucciones de crear una LAN más rápida. Una propuesta fue mantener 802.3 exactamente como estaba, pero hacerla más rápida. Otra propuesta fue rehacerla en su totalidad para darle muchas características nuevas, como tráfico en tiempo real y voz digitalizada, pero mantener el nombre antiguo (por razones de marketing). Después de algunas discusiones, el comité decidió mantener la Ethernet 802.3 tal como estaba, pero hacerla más rápida. Las personas que apoyaban la propuesta contraria hicieron lo que cualquier persona de la industria de la computación habría hecho bajo estas circunstancias —unieron fuerzas y formaron su propio comité y estandarizaron su LAN de todas maneras (que con el tiempo se llamó 802.12). Ésta fracasó rotundamente.

El comité 802.3 decidió crear una Ethernet mejorada por tres razones principales:

1. La necesidad de compatibilidad hacia atrás con las LANs Ethernet existentes.
2. El miedo de que un nuevo protocolo tuviera problemas no previstos.
3. El deseo de terminar el trabajo antes de que la tecnología cambiara.

El trabajo se terminó rápidamente (mediante las normas de los comités de estándares), y el resultado, **802.3u,** fue aprobado oficialmente por el IEEE en junio de 1995. Técnicamente, 802.3u no es un nuevo estándar, sino un agregado al estándar existente 802.3 (para enfatizar su compatibilidad hacia atrás). Puesto que prácticamente todos lo llaman **Fast Ethernet,** en lugar de 802.3u, nosotros también lo haremos.

La idea básica detrás de Fast Ethernet era sencilla: mantener todos los formatos anteriores, interfaces y reglas de procedimientos, y sólo reducir el tiempo de bits de 100 nseg a 10 nseg. Técnicamente, habría sido posible copiar 10Base-5 o 10Base-2 y aún detectar colisiones a tiempo con sólo reducir la longitud máxima de cable por un factor de diez. Sin embargo, las ventajas del cableado 10Base-T eran tan abrumadoras que Fast Ethernet se basa por completo en este diseño. Por lo tanto, todos los sistemas Fast Ethernet utilizan concentradores y conmutadores; no se permiten cables con múltiples derivaciones vampiro ni conectores BNC.

Sin embargo, aún se tienen que tomar algunas decisiones, la más importante de las cuáles es qué tipos de cable soportar. Un contendiente era el cable de par trenzado categoría 3. El argumento a su favor era que prácticamente todas las oficinas en el mundo occidental tienen por lo menos cuatro cables de par trenzado categoría tres (o mejor) que van de la oficina hacia un gabinete de cableado telefónico dentro de una distancia de 100 metros. Algunas veces existen dos de esos cables. Por lo tanto, el uso del cable de par trenzado categoría 3 hace posible cablear las computadoras de escritorio mediante Fast Ethernet sin tener que volver a cablear el edificio, lo cual es una enorme ventaja para muchas organizaciones.

La principal desventaja del cable de par trenzado categoría 3 es su incapacidad de llevar señales de 200 megabaudios (100 Mbps con codificación Manchester) a una distancia de hasta100 metros, que es la distancia máxima de computadora a concentrador especificada para 10Base-T (vea la figura 4-13). En contraste, el cable de par trenzado categoría 5 puede manejar 100 metros con facilidad, y la fibra puede ir mucho más rápido. El arreglo elegido fue permitir las tres posibilidades, como se muestra en la figura 4-21, pero fortalecer la solución categoría 3 para darle la capacidad de transmisión adicional necesaria.

Nombre	Cable	Segmento máximo	Ventajas
100Base-T4	Par trenzado	100 m	Utiliza UTP categoría 3
100Base-TX	Par trenzado	100 m	Dúplex total a 100 Mbps (UTP cat 5)
100Base-FX	Fibra óptica	2000 m	Dúplex total a 100 Mbps; distancias largas

Figura 4-21. El cableado original de Fast Ethernet.

El esquema UTP categoría 3, llamado **100Base-T4,** utiliza una velocidad de señalización de 25 MHz, tan sólo 25 por ciento más rápida que los 20 MHz de la Ethernet estándar (recuerde que la codificación Manchester, como se muestra en la figura 4-16, requiere dos periodos de reloj para cada uno de los 10 millones de bits cada segundo). Sin embargo, para alcanzar el ancho de banda necesario, 100Base-T4 requiere cuatro cables de par trenzado. Debido a que el cableado telefónico estándar durante décadas ha tenido cuatro cables de par trenzado por cable, la mayoría de las oficinas puede manejar esto. Por supuesto, esto significa ceder su teléfono de la oficina, pero seguramente eso es un precio pequeño a cambio de correo electrónico más rápido.

De los cuatro cables de par trenzado, uno siempre va al concentrador, uno siempre sale del concentrador y los otros dos son intercambiables a la dirección actual de transmisión. Para obtener el ancho de banda necesario, no se utiliza la codificación Manchester, pero con relojes modernos y distancias cortas, ya no es necesaria. Además, se envían señales ternarias, para que durante un periodo de reloj el cable pueda contener un 0, un 1 o un 2. Con tres cables de par trenzado y la señalización ternaria, se puede transmitir cualquiera de 27 símbolos posibles, con lo que se pueden enviar 4 bits con algo de redundancia. Transmitir 4 bits en cada uno de los 25 millones de ciclos de reloj por segundo da los 100 Mbps necesarios. Además, siempre hay un canal de regreso de 33.3 Mbps que utiliza el resto del cable de par trenzado. No es probable que este esquema, conocido como **8B/6T** (8 bits se convierten en 6 trits), gane un premio por elegancia, pero funciona con la planta de cableado existente.

Para el cableado categoría 5, el diseño **100Base-TX** es más simple porque los cables pueden manejar velocidades de reloj de 125 MHz. Sólo se utilizan dos cables de par trenzado por estación, uno para enviar y otro para recibir. La codificación binaria directa no se utiliza; en su lugar se toma un esquema llamado **4B/5B** tomado de las redes FDDI, y compatible con ellas. Cada grupo de cinco periodos de reloj, cada uno de los cuales contiene uno de dos valores de señal, da 32 combinaciones. Dieciséis de estas combinaciones se utilizan para transmitir los cuatro grupos de bits 0000, 0001, 0010, ..., 1111. Algunos de los 16 restantes se utilizan para propósitos de control, como el marcado de límites de tramas. Las combinaciones utilizadas se han elegido cuidadosamente para proporcionar suficientes transiciones para mantener sincronización de reloj. El sistema 100Base-TX es de dúplex total; las estaciones pueden transmitir a 100 Mbps y recibir a 100 Mbps al mismo tiempo. Con frecuencia, 100Base-TX y 100Base-T4 se llaman en conjunto **100Base-T.**

La última opción, **100Base-FX,** utiliza dos filamentos de fibra multimodo, una para cada dirección, por lo que también es dúplex total con 100 Mbps en cada dirección. Además, la distancia entre una estación y el concentrador puede ser de hasta 2 km.

En respuesta a la demanda popular, en 1997 el comité 802 agregó un nuevo tipo de cableado, 100Base-T2, que permite que la Fast Ethernet se ejecute a través de dos pares de cables existentes de categoría 3. Sin embargo, se necesita un procesador de señales digital sofisticado para manejar el esquema de codificación requerido, lo que hace de esta opción algo muy costoso. Hasta ahora su uso es muy inusual debido a su complejidad y costo, así como al hecho de que muchos edificios de oficinas se han vuelto a cablear con UTP categoría 5.

Con 100Base-T son posibles dos tipos de dispositivos de interconexión: concentradores y conmutadores, como se muestra en la figura 4-20. En un concentrador, todas las líneas entrantes (o al menos todas las líneas que llegan a una tarjeta de conexión) se conectan lógicamente, formando

un solo dominio de colisión. Se aplican todas las reglas estándar, entre ellas el algoritmo de retroceso exponencial binario, por lo que el sistema funciona de la misma manera que la Ethernet antigua. En particular, sólo una estación a la vez puede transmitir. En otras palabras, los concentradores requieren comunicación semidúplex.

En un conmutador, cada trama entrante se almacena en el búfer de una tarjeta de conexión y se pasa a través de una matriz de conmutación de alta velocidad de la tarjeta de origen a la de destino, si es necesario. La matriz de conmutación no se ha estandarizado, ni lo necesita, debido a que está completamente oculta dentro del conmutador. Si la experiencia pasada sirve de algo, los fabricantes de conmutadores competirán con ardor para producir matrices de conmutación más veloces para mejorar la velocidad real de transporte del sistema. Debido a que los cables 100Base-FX son muy largos para el algoritmo de colisiones de la Ethernet, deben conectarse a conmutadores, de manera que cada uno sea un dominio de colisión en sí mismo. Los concentradores no están permitidos con 100Base-FX.

Como nota final, casi todos los conmutadores pueden manejar una mezcla de estaciones de 10 y 100 Mbps, para facilitar la actualización. Conforme un sitio adquiera más y más estaciones de trabajo de 100 Mbps, todo lo que tiene que hacer es comprar la cantidad necesaria de tarjetas de línea e insertarlas en el conmutador. De hecho, el estándar mismo proporciona una forma para que dos estaciones negocien de manera automática la velocidad óptima (10 o 100 Mbps) y el tipo de transmisión dúplex (semi o total). La mayoría de los productos de Fast Ethernet utilizan esta característica para autoconfigurarse.

4.3.8 Gigabit Ethernet

La tinta apenas se estaba secando en el estándar de la Fast Ethernet cuando el comité 802 comenzó a trabajar en una Ethernet aún más rápida (1995). Se conoció como **Gigabit Ethernet** y fue aprobada por el IEEE en 1998 bajo el nombre 802.3z. Este identificador sugiere que la Gigabit Ethernet va a ser el final de la línea, a menos que alguien invente rápidamente una letra después de la z. A continuación analizaremos algunas de las características principales de la Gigabit Ethernet. Para mayor información, vea (Seifert, 1998).

Los objetivos del comité 802.3z eran esencialmente los mismos que los del comité 802.3u: hacer que Ethernet fuera 10 veces más rápida y que permaneciera compatible hacia atrás con todos los estándares Ethernet existentes. En particular, Gigabit Ethernet tiene que ofrecer servicio de datagramas sin confirmación de recepción con difusión y multidifusión, utilizar el mismo esquema de direccionamiento de 48 bits que el actual y mantener el mismo formato de trama, incluyendo los tamaños mínimo y máximo de trama. El estándar final cumple con todos estos objetivos.

Todas las configuraciones de Gigabit Ethernet son de punto a punto en lugar de múltiples derivaciones como en el estándar original de 10 Mbps, ahora conocido como **Ethernet clásica.** En la configuración más simple de Gigabit Ethernet, que se muestra en la figura 4-22(a), dos computadoras están conectadas de manera directa entre sí. Sin embargo, el caso más común es tener un conmutador o un concentrador conectado a múltiples computadoras y posiblemente a conmutadores o concentradores adicionales, como se muestra en la figura 4-22(b). En ambas configuraciones cada cable Ethernet individual tiene exactamente dos dispositivos en él, ni más ni menos.

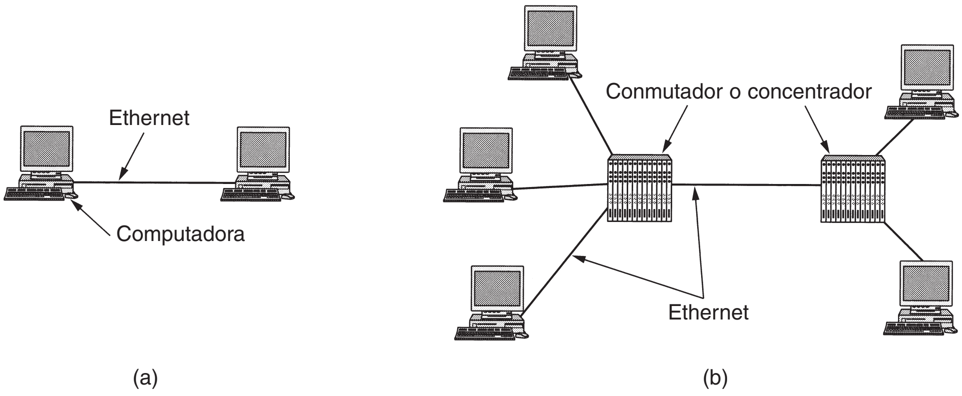


Figura 4-22. (a) Ethernet de dos estaciones. (b) Ethernet con múltiples estaciones.

Gigabit Ethernet soporta dos modos diferentes de funcionamiento: modo de dúplex total y modo de semidúplex. El modo “normal” es el de dúplex total, el cual permite tráfico en ambas direcciones al mismo tiempo. Este modo se utiliza cuando hay un conmutador central conectado a computadoras (o a otros conmutadores) en el periférico. En esta configuración, todas las líneas se almacenan en el búfer a fin de que cada computadora y conmutador pueda enviar tramas siempre que lo desee. El emisor no tiene que detectar el canal para ver si alguien más lo está utilizando debido a que la contención es imposible. En la línea entre una computadora y un conmutador, la computadora es el único emisor posible en esa línea al conmutador y la transmisión tiene éxito aun cuando el conmutador esté enviado actualmente una trama a la computadora (porque la línea es de dúplex total). Debido a que no hay contención, no se utiliza el protocolo CSMA/CD y la longitud máxima del cable se determina con base en la fuerza de la señal más que en el tiempo que tarda una ráfaga de ruido en regresar al emisor en el peor caso. Los conmutadores son libres de mezclar e igualar velocidades. La autoconfiguración se soporta al igual que en Fast Ethernet.

El otro modo de operación, semidúplex, se utiliza cuando las computadoras están conectadas a un concentrador en lugar de a un conmutador. Un concentrador no almacena en el búfer las tramas entrantes. En su lugar, conecta en forma eléctrica todas las líneas internamente, simulando el cable con múltiples derivaciones que se utiliza en la Ethernet clásica. En este modo las colisiones son posibles, por lo que es necesario el protocolo CSMA/CD estándar. Debido a que una trama mínima (de 64 bytes) ahora puede transmitirse 100 veces más rápido que en la Ethernet clásica, la distancia máxima es 100 veces menor, o 25 metros, para mantener la propiedad esencial de que el emisor aún transmita cuando la ráfaga de ruido vuelva a él, incluso en el peor caso. Con un cable de 2500 metros de longitud, el emisor de una trama de 64 bytes a 1 Gbps podría terminar su transmisión antes de que la trama recorra una décima del camino, y muchísimo antes de que llegue al otro extremo y regrese.

El comité 802.3z consideró un radio de 25 metros como inaceptable y agregó dos características al estándar para incrementar el radio. La primera, llamada **extensión de portadora,** esencialmente indica al hardware que agregue su propio relleno después de la trama normal para extenderla a 512 bytes. Puesto que este relleno es agregado por el hardware emisor y eliminado

por el hardware receptor, el software no toma parte en esto, lo que significa que no es necesario realizar cambios al software existente. Por supuesto, utilizar 512 bytes de ancho de banda para transmitir 46 bytes de datos de usuario (la carga útil de una trama de 64 bytes) tiene una eficiencia de línea de 9%.

La segunda característica, llamada **ráfagas de trama,** permite que un emisor transmita una secuencia concatenada de múltiples tramas en una sola transmisión. Si la ráfaga total es menor que 512 bytes, el hardware la rellena nuevamente. Si suficientes tramas están esperando la transmisión, este esquema es muy eficiente y se prefiere antes que la extensión de portadora. Estas nuevas características amplían el radio de red de 200 metros, que probablemente es suficiente para la mayoría de las oficinas.

Sin duda, una organización difícilmente enfrentará el problema de comprar e instalar tarjetas Gigabit Ethernet para obtener mayor rendimiento y después conectar las computadoras con un concentrador para simular una Ethernet clásica con todas sus colisiones. Si bien los concentradores son un tanto más baratos que los conmutadores, las tarjetas de interfaz Gigabit Ethernet aún son relativamente costosas. Por lo tanto, tratar de economizar comprando un concentrador barato y reducir drásticamente el desempeño del nuevo sistema es algo impensable. Además, la compatibilidad hacia atrás es sagrada en la industria de la computación, por lo que se le pidió al comité 802.3z que la añadiera.

Como se lista en la figura 4-23, Gigabit Ethernet soporta tanto el cableado de fibra óptica como el de cobre. Transmitir señales a o aproximadamente a 1 Gbps a través de fibra significa que la fuente de luz debe encenderse y apagarse en 1 nseg. Los LEDs simplemente no pueden funcionar con esta rapidez, por lo que se necesitan láseres. Se permiten dos longitudes de onda: 0.85 micras (Corto) y 1.3 micras (Largo). Los láseres a 0.85 micras son más económicos pero no funcionan en una fibra de modo sencillo.

Nombre	Cable	Segmento máximo	Ventajas
1000Base-SX	Fibra óptica	550 m	Fibra multimodo (50, 62.5 micras)
1000Base-LX	Fibra óptica	5000 m	Sencilla (10 μ) o multimodo (50, 62.5 μ)
1000Base-CX	2 pares de STP	25 m	Cable de par trenzado blindado
1000Base-T	4 Pares de UTP	100 m	UTP categoría 5 estándar

Figura 4-23. Cableado de Gigabit Ethernet.

Se permiten tres diámetros de fibra: 10, 50 y 62.5 micras. El primero es para el modo sencillo y los últimos dos son para multimodo. Sin embargo, no se permiten las seis combinaciones y la distancia máxima depende de la combinación que se utilice. Los números que se dan en la figura 4-23 son para el mejor de los casos. En particular, 5000 metros son alcanzables únicamente con láseres de 1.3 micras que funcionen a través de fibra de 10 micras en modo sencillo, pero ésta es la mejor opción para redes dorsales y se espera que sea popular, a pesar de ser la opción más costosa.

La opción 1000Base-CX utiliza cables de cobre blindados cortos. Su problema es que compite con la fibra de alto desempeño por una parte, y con el UTP más económico por la otra. No es probable que se utilice mucho, si es que se utiliza.

La última opción son paquetes de cuatro cables UTP categoría 5 que trabajan juntos. Debido a que la mayor parte de este cableado ya está instalada, es probable que sea la Gigabit Ethernet de la gente pobre.

Gigabit Ethernet utiliza nuevas reglas de codificación en las fibras. La codificación Manchester a 1 Gbps podría requerir una señal de 2 Gbaudios, lo cual era considerado muy difícil y también un desperdicio de ancho de banda. En su lugar se eligió un nuevo esquema, llamado **8B/10B,** que se basa en un canal de fibra. Cada byte de 8 bits está codificado en la fibra como 10 bits, de aquí el nombre 8B/10B. Debido a que hay 1024 palabras codificadas posibles para cada byte de entrada, hay algo de libertad al elegir cuáles palabras codificadas permitir. Las siguientes dos reglas se utilizaron al realizar las elecciones:

1. Ninguna palabra codificada podría tener más de cuatro bits idénticos en una fila.
2. Ninguna palabra codificada podría tener más de seis bits 0 o seis bits 1.

Estas elecciones se realizaron para mantener suficientes transiciones en el flujo para asegurarse de que el receptor continúe sincronizado con el emisor y también para mantener la cantidad de bits 0 y bits 1 en la fibra tan cerca del equilibrio como sea posible. Además, muchos bytes de entrada tienen dos palabras codificadas posibles asignadas a ellos. Cuando el codificador tiene la opción de palabras codificadas, siempre elige la palabra codificada que tiende a igualar la cantidad de 0s y 1s transmitidos hasta ese momento. Este énfasis en igualar 0s y 1s es necesario para mantener el componente DC de la señal tan bajo como sea posible para permitirle pasar a través de transformadores no modificados. Si bien los científicos de la computación no son afectos a que las propiedades de los transformadores dicten sus esquemas de codificación, algunas veces la vida es así.

Las Gigabit Ethernet que utilizan 1000Base-T emplean un esquema de codificación diferente debido a que cronometrar el tiempo de los datos en el cable de cobre en 1 nseg es muy difícil. Esta solución utiliza cuatro cables de par trenzado categoría 5 para permitir que se transmitan en paralelo cuatro símbolos. Cada uno de ellos se codifica utilizando uno de cinco niveles de voltaje. Este esquema permite que un solo símbolo codifique 00, 01, 10, 11 o un valor especial para propósitos de control. Por lo tanto, hay 2 bits de datos por cable de par tranzado u 8 bits de datos por ciclo de reloj. El reloj se ejecuta a 125 MHz, y permite una operación de 1 Gbps. La razón para permitir cinco niveles de voltaje en lugar de cuatro es tener combinaciones sobrantes para propósitos de entramado y control.

1 Gbps es una velocidad muy alta. Por ejemplo, si un receptor está ocupado con otra tarea por incluso un 1 mseg y no vacía el búfer de entrada en alguna línea, podrían haberse acumulado ahí hasta 1953 tramas en ese espacio de 1 ms. Además, cuando una computadora en una Gigabit Ethernet está enviando datos en la línea a una computadora en una Ethernet clásica, es muy probable que sucedan rebases de búfer. Como consecuencia de estas dos observaciones, Gigabit Ethernet soporta control de flujo (como lo hace la Fast Ethernet, aunque los dos son diferentes).

El control de flujo consiste en que un extremo envíe una trama de control especial al otro extremo indicándole que se detenga por algún tiempo. Las tramas de control son tramas comunes de Ethernet que contienen un tipo de 0x8808. Los primeros dos bytes del campo de datos dan el comando; los bytes exitosos proporcionan los parámetros, si es que hay. Para control de flujo, se utilizan las tramas PAUSE, en las que el parámetro indica cuánto tiempo detenerse, en unidades de tiempo de la trama más pequeña. Para la Gigabit Ethernet, la unidad de tiempo es 512 nseg, lo que permite pausas de 33.6 mseg.

Una vez que la Gigabit Ethernet se estandarizó, el comité 802 se aburrió y quiso volver al trabajo. El IEEE les dijo que iniciaran una Ethernet de 10 gigabits. Después de buscar arduamente una letra que siguiera a la z, el comité abandonó ese enfoque y se concentró en los sufijos de dos letras. Comenzó el trabajo y ese estándar fue aprobado por el IEEE en el 2002 como 802.3ae. ¿Es posible que le siga una Ethernet de 100 gigabits?

4.3.9. Estándar IEEE 802.2: control lógico del enlace

Tal vez ahora sea el momento de dar un paso atrás y comparar lo que hemos aprendido en este capítulo con lo que estudiamos en el anterior. En el capítulo 3 vimos la manera en que dos máquinas se podían comunicar de manera confiable a través de una línea inestable usando varios protocolos de enlace de datos. Estos protocolos proporcionaban control de errores (mediante confirmaciones de recepción) y control de flujo (usando una ventana corrediza).

En contraste, en este capítulo no hemos mencionado las comunicaciones confiables. Todo lo que ofrecen las Ethernet y los protocolos 802 es un servicio de datagramas de mejor esfuerzo. A veces, este servicio es adecuado. Por ejemplo, para transportar paquetes IP no se requieren ni se esperan garantías. Un paquete IP simplemente puede introducirse en un campo de carga 802 y enviarse a su destino; si se pierde, que así sea.

Sin embargo, también hay sistemas en los que se desea un protocolo de enlace de datos con control de errores y control de flujo. El IEEE ha definido uno que puede operar encima de todos los protocolos Ethernet y 802. Además, este protocolo, llamado **LLC** (**Control Lógico del Enlace**), esconde las diferencias entre los distintos tipos de redes 802, proporcionando un formato único y una interfaz con la capa de red. Este formato, interfaz y protocolo están basados estrechamente en HDLC que estudiamos en el capítulo 3. El LLC forma la mitad superior de la capa de enlace de datos, con la subcapa de MAC por debajo de él, como se muestra en la figura 4-24.

El uso típico del LLC es el siguiente. La capa de red de la máquina emisora pasa un paquete al LLC usando las primitivas de acceso del LLC. A continuación, la subcapa LLC agrega un encabezado LLC que contiene los números de secuencia y confirmación de recepción. La estructura resultante se introduce entonces en el campo de carga útil de una trama 802 y se transmite. En el receptor ocurre el proceso inverso.

El LLC proporciona tres opciones de servicio: servicio no confiable de datagramas, servicio de datagramas sin confirmación de recepción y servicio confiable orientado a la conexión. El encabezado LLC contiene tres campos: un punto de acceso de destino, un punto de acceso de origen y un campo de control. Los puntos de acceso indican de cuál proceso proviene la trama y en dónde

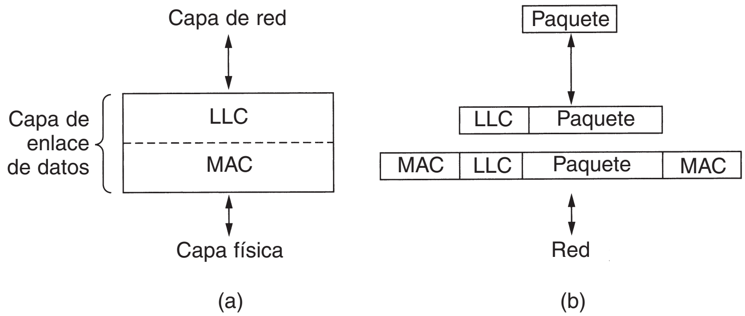


Figura 4-24. (a) Posición del LLC. (b) Formatos de protocolo.

se va a enviar, con lo que reemplazan el campo de *Tipo* DIX. El campo de control contiene números de secuencia y de confirmación de recepción, muy parecido a HDLC (vea la figura 3-24), pero no idéntico. Estos campos se utilizan principalmente cuando se necesita una conexión confiable en el nivel de enlace de datos, en cuyo caso se utilizarían protocolos similares a los que tratamos en el capítulo 3. Para Internet, los intentos de mejor esfuerzo para enviar los paquetes IP son suficientes, por lo que no se requieren confirmaciones de recepción en el nivel LLC.

4.3.10 Retrospectiva de Ethernet

Ethernet ha existido desde hace 20 años y no tiene competidores serios a la vista, por lo que es probable que exista por algunos años más. Pocas arquitecturas de CPU, sistemas operativos o lenguajes de programación han sido los reyes de la montaña por más de dos décadas. Claramente, Ethernet hizo algo bien, pero, ¿qué fue?

Probablemente la razón principal de su longevidad es que Ethernet es simple y flexible. En la práctica, simple se traduce como confiable, barato y fácil de mantener. Una vez que las derivaciones vampiro se reemplazaron con conectores BNC, las fallas eran menos frecuentes. Las personas dudaban en reemplazar algo que funcionaba bien todo el tiempo, especialmente porque sabían que muchas cosas funcionaban pobremente en la industria de la computación, por lo que muchas "actualizaciones" son peores que lo que reemplazan.

Simple también se traduce como barato. El cableado Ethernet delgado y el de par trenzado tienen un costo relativamente bajo. Las tarjetas de interfaz también tienen un costo bajo. Sólo cuando se introdujeron concentradores y conmutadores, se necesitaron inversiones considerables, pero para la época en que entraron en escena, Ethernet ya estaba bien establecida.

Ethernet es fácil de mantener. No hay software que instalar (sólo los controladores) y no hay tablas de configuración que manejar (con las cuales equivocarse). Además, agregar nuevos *hosts* es tan simple como conectarlos.

Otro punto es que Ethernet interactúa fácilmente con TCP/IP, el cual se ha vuelto dominante. IP es un protocolo sin conexión, porque se ajusta perfectamente con Ethernet, que tampoco es orientado a la conexión. IP no se ajusta tan bien con ATM, que es orientado a la conexión. Esta falta de ajuste afecta definitivamente las posibilidades de ATM.

Por último, Ethernet ha sido capaz de evolucionar en formas importantes. Las velocidades han aumentado en algunos niveles de magnitud y se han introducido los concentradores y conmutadores, pero estos cambios no requieren modificaciones en el software. Un vendedor de redes está en un grave problema cuando muestra una instalación grande y dice: "Tengo esta nueva red fantástica para usted. Lo único que tiene que hacer es tirar todo su hardware y reescribir todo su software". Cuando se introdujeron la FDDI, el canal de fibra y ATM, eran más rápidos que Ethernet, pero también eran incompatibles con Ethernet, mucho más complejos y difíciles de manejar. Con el tiempo, Ethernet los igualó en cuanto a velocidad, por lo que ya no tenían ventajas y poco a poco dejaron de utilizarse, excepto ATM, el cual se utiliza en el núcleo del sistema telefónico.

4.4 LANS INALÁMBRICAS

Aunque Ethernet se utiliza ampliamente, está a punto de tener un competidor fuerte. Las LANs inalámbricas se están volviendo muy populares, y más y más edificios de oficinas, aeropuertos y otros lugares públicos se están equipando con ellas. Las LANs inalámbricas pueden funcionar en una de dos configuraciones, como vimos en la figura 1-35: con una estación base y sin ninguna estación base. En consecuencia, el estándar de LAN 802.11 toma en cuenta esto y se previene para ambos arreglos, como veremos más adelante.

En la sección 1.5.4 proporcionamos información sobre 802.11. Ahora es tiempo de ver más de cerca esta tecnología. En las siguientes secciones veremos la pila de protocolos, las técnicas de transmisión de radio de la capa física, el protocolo de la subcapa MAC, la estructura de trama y los servicios. Para mayor información sobre 802.11, vea (Crow y cols., 1997; Geier, 2002; Heegard y cols., 2001; Kapp, 2002; O'Hara y Petrick, 1999, y Severance, 1999). Para obtener información de una fuente fidedigna, consulte el estándar publicado 802.11.

4.4.1 La pila de protocolos del 802.11

Los protocolos utilizados por todas las variantes 802, entre ellas Ethernet, tienen ciertas similitudes de estructura. En la figura 4-25 se muestra una vista parcial de la pila de protocolos del estándar 802.11. La capa física corresponde muy bien con la capa física OSI, pero la capa de enlace de datos de todos los protocolos 802 se divide en dos o más subcapas. En el estándar 802.11, la subcapa MAC determina la forma en que se asigna el canal, es decir, a quién le toca transmitir a continuación. Arriba de dicha subcapa se encuentra la subcapa LLC, cuyo trabajo es ocultar las diferencias entre las variantes 802 con el propósito de que sean imperceptibles para la capa de red. Anteriormente en este capítulo analizamos el LLC, cuando examinamos Ethernet, por lo que no repetiremos ese material aquí.

El estándar 802.11 de 1997 especifica tres técnicas de transmisión permitidas en la capa física. El método de infrarrojos utiliza en su mayor parte la misma tecnología que los controles remotos

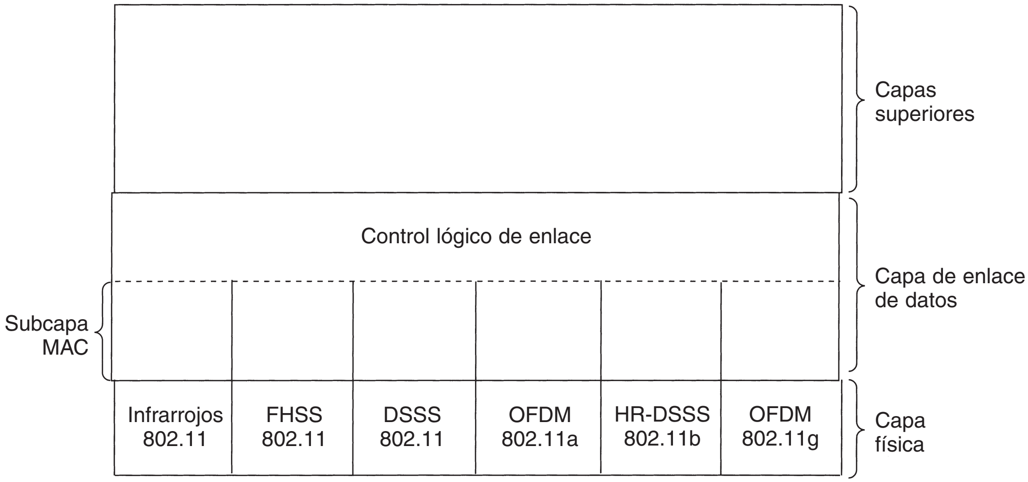


Figura 4-25. Parte de la pila de protocolos del 802.11.

de televisión. Los otros dos métodos utilizan el radio de corto alcance, mediante técnicas conocidas como FHSS y DSSS. Éstas utilizan parte del espectro que no necesita licencia (la banda ISM de 2.4 GHz). Los abridores de puertas de cocheras controlados por radio también utilizan esta parte del espectro, por lo que su computadora portátil podría encontrarse compitiendo con la puerta de la cochera. Los teléfonos inalámbricos y los hornos de microondas también utilizan esta banda. Todas estas técnicas funcionan a 1 o 2 Mbps y con poca energía por lo que no interfieren mucho entre sí. En 1999 se introdujeron dos nuevas técnicas para alcanzar un ancho de banda más alto. Éstas se conocen como OFDM y HRDSSS. Funcionan hasta 54 y 11 Mbps, respectivamente. En 2001 se introdujo una segunda modulación OFDM, pero en una banda de frecuencia diferente respecto a la primera. A continuación examinaremos con brevedad cada una de ellas. Técnicamente, pertenecen a la capa física y debieron examinarse en el capítulo 2, pero las trataremos aquí debido a que están estrechamente enlazadas a las LANs en general y a la subcapa MAC 802.11.

4.4.2 La capa física del 802.11

Cada una de las cinco técnicas permitidas de transmisión posibilitan el envío de una trama MAC de una estación a otra. Sin embargo, difieren en la tecnología utilizada y en las velocidades alcanzables. Un análisis detallado de estas tecnologías está más allá del alcance de este libro, pero es posible que algunas palabras sobre dichas tecnologías, junto con algunos términos clave, proporcionen a los lectores interesados algunos términos con los cuales buscar más información en Internet o en alguna otra parte.

La opción de infrarrojos utiliza transmisión difusa (es decir, no requiere línea visual) a 0.85 o 0.95 micras. Se permiten dos velocidades: 1 y 2 Mbps. A 1 Mbps se utiliza un esquema de codificación en el cual un grupo de 4 bits se codifica como una palabra codificada de 16 bits, que contiene quince 0s y un 1, mediante **código de Gray.** Este código tiene la propiedad de que un pequeño error en la sincronización en el tiempo lleva a un solo error de bits en la salida. A 2 Mbps, la codificación toma 2 bits y produce una palabra codificada de 4 bits, también con un solo 1, que es uno de 0001, 0010, 0100 o 1000. Las señales de infrarrojos no pueden penetrar las paredes, por lo que las celdas en los diferentes cuartos están bien aisladas entre sí. Sin embargo, debido al bajo ancho de banda (y al hecho de que la luz solar afecta las señales de infrarrojos), ésta no es una opción muy popular.

FHSS (**Espectro Disperso con Salto de Frecuencia**) utiliza 79 canales, cada uno de los cuales tiene un ancho de banda de 1 MHz, iniciando en el extremo más bajo de la banda ISM de 2.4 GHz. Para producir la secuencia de frecuencias a saltar, se utiliza un generador de números pseudoaleatorios. Siempre y cuando todas las estaciones utilicen la misma semilla para el generador de números pseudoaleatorios y permanezcan sincronizadas, saltarán de manera simultánea a la misma frecuencia. El tiempo invertido en cada frecuencia, el **tiempo de permanencia,** es un parámetro ajustable, pero debe ser menor que 400 mseg. La aleatorización de FHSS proporciona una forma justa de asignar espectro en la banda ISM no regulada. También proporciona algo de seguridad pues un intruso que no sepa la secuencia de saltos o el tiempo de permanencia no puede espiar las transmisiones. En distancias más grandes, el desvanecimiento de múltiples rutas puede ser un problema, y FHSS ofrece buena resistencia a ello. También es relativamente insensible a la interferencia de radio, lo que lo hace popular para enlaces de edificio en edificio. Su principal desventaja es su bajo ancho de banda.

El tercer método de modulación, **DSSS** (**Espectro Disperso de Secuencia Directa**)**,** también está restringido a 1 o 2 Mbps. El esquema utilizado tiene algunas similitudes con el sistema CDMA que examinamos en la sección 2.6.2, pero difiere en otros aspectos. Cada bit se transmite como 11 *chips*, utilizando lo que se conoce como **secuencia Barker.** Utiliza modulación por desplazamiento de fase a 1 Mbaudio, y transmite 1 bit por baudio cuando opera a 1 Mbps, y 2 bits por baudio cuando opera a 2 Mbps. Durante mucho tiempo, la FCC exigió que todo el equipo de comunicación inalámbrica que operaba en la banda ISM en Estado Unidos utilizará el espectro disperso, pero en mayo de 2002 esa regla se eliminó conforme apareció nueva tecnología.

La primera de las LANs inalámbricas de alta velocidad, **802.11a,** utiliza **OFDM** (**Multiplexión por División de Frecuencias Ortogonales**) para enviar hasta 54 Mbps en la banda ISM más ancha de 5 GHz. Como lo sugiere el término FDM, se utilizan frecuencias diferentes —52 en total, 48 para datos y 4 para sincronización— al igual que ADSL. Debido a que las transmisiones están presentes en múltiples frecuencias al mismo tiempo, esta técnica se considera como una forma de espectro disperso, pero es diferente a CDMA y a FHSS. Dividir la señal en bandas más estrechas tiene más ventajas que el uso de una sola banda ancha, entre ellas mejor inmunidad a la interferencia de bandas estrechas y la posibilidad de utilizar bandas no contiguas. Se utiliza un sistema de codificación complejo, con base en la modulación por desplazamiento de fase para velocidades de hasta 18 Mbps, y en QAM para velocidades mayores. A 54 Mbps, se codifican 216 bits

de datos en símbolos de 288 bits. Parte del motivo para utilizar OFDM es la compatibilidad con el sistema europeo HiperLAN/2 (Doufexi y cols., 2002). La técnica tiene buena eficiencia de espectro en términos de bits/Hz y buena inmunidad al desvanecimiento de múltiples rutas.

A continuación analizaremos **HR-DSSS** (**Espectro Disperso de Secuencia Directa de Alta Velocidad**), otra técnica de espectro disperso, que utiliza 11 millones de chips/seg para alcanzar 11 Mbps en la banda de 2.4 GHz. Se llama **802.11b** pero no es la continuación de 802.11a. De hecho, su estándar se aprobó primero y apareció primero en el mercado. Las tasas de datos soportadas por 802.11b son 1, 2, 5.5 y 11 Mbps. Las dos tasas bajas se ejecutan a 1 Mbaudio, con 1 y 2 bits por baudio, respectivamente, utilizando modulación por desplazamiento de fase (por compatibilidad con DSSS). Las dos tasas más rápidas se ejecutan a 1.375 Mbaudios, con 4 y 8 bits por baudio, respectivamente, utilizando códigos **Walsh/Hadamard.** La tasa de datos puede ser adaptada de manera dinámica durante la operación para alcanzar la velocidad más óptima posible bajo las condiciones actuales de la carga y el ruido. En la práctica, la velocidad de operación de 802.11b siempre es de aproximadamente 11 Mbps. Aunque 802.11b es más lento que 802.11a, su rango es aproximadamente 7 veces mayor, lo que es más importante en muchas situaciones.

En noviembre de 2001, el IEEE aprobó una versión mejorada de 802.11b, **802.11g,** después de mucho politiqueo por cuál tecnología patentada podría utilizar. Utiliza el método de modulación OFDM de 802.11a pero opera en la banda ISM más estrecha 2.4 GHz ISM junto con 802.11b. En teoría, puede operar hasta a 54 Mbps. Aún no se ha decidido si esta velocidad se va a alcanzar en la práctica. Lo que esto significa es que el comité 802.11 ha producido tres LANs inalámbricas diferentes de alta velocidad: 802.11a, 802.11b y 802.11g (sin mencionar las tres LANs inalámbricas de baja velocidad). Uno se puede preguntar si es bueno que el comité de estándares haga esto. Tal vez el tres sea su número de suerte.

4.4.3 El protocolo de la subcapa MAC del 802.11

Regresemos ahora de la tierra de la ingeniería eléctrica a la de las ciencias de la computación. El protocolo de la subcapa MAC para el estándar 802.11 es muy diferente del de Ethernet debido a la complejidad inherente del entorno inalámbrico en comparación con el de un sistema cableado. Con Ethernet, una estación simplemente espera hasta que el medio queda en silencio y comienza a transmitir. Si no recibe una ráfaga de ruido dentro de los primeros 64 bytes, con seguridad la trama ha sido entregada correctamente. Esta situación no es válida para los sistemas inalámbricos.

Para empezar, existe el problema de la estación oculta mencionado con anterioridad, el cual se ilustra nuevamente en la figura 4-26(a). Puesto que no todas las estaciones están dentro del alcance de radio de cada una, las transmisiones que van en un lado de una celda podrían no recibirse en otro lado de la misma celda. En este ejemplo, la estación *C* transmite a la estación *B*. Si *A* detecta el canal, no escuchará nada y concluirá erróneamente que ahora puede comenzar a transmitir a *B*.

Además, existe el problema inverso, el de la estación expuesta, que se ilustra en la figura 4-26(b). Aquí *B* desea enviar a *C* por lo que escucha el canal. Cuando escucha una transmisión, concluye erróneamente que no debería transmitir a *C*, aunque *A* esté transmitiendo a *D* (lo cual no

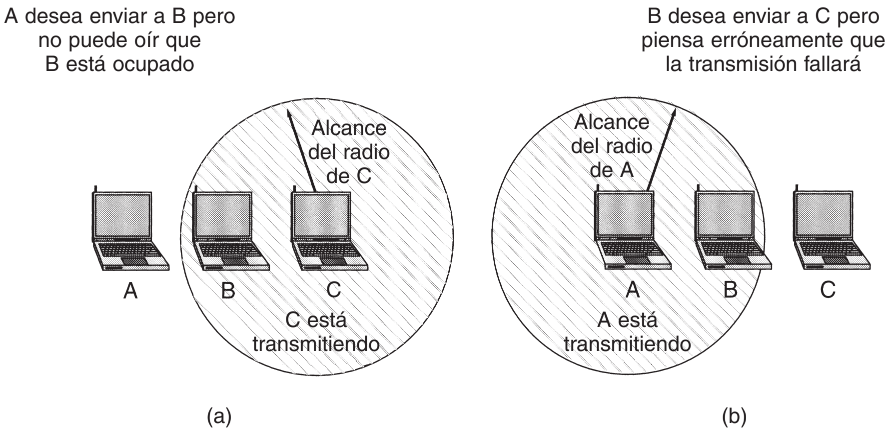


Figura 4-26. (a) El problema de la estación oculta. (b) El problema de la estación expuesta.

se muestra). Además, la mayoría de los radios son semidúplex, lo que significa que no pueden transmitir y escuchar ráfagas de ruido al mismo tiempo en una sola frecuencia. Como resultado de estos problemas, 802.11 no utiliza CSMA/CD, como lo hace Ethernet.

Para solucionar este problema, 802.11 suporta dos modos de funcionamiento. El primero, llamado **DCF** (**Función de Coordinación Distribuida),** no utiliza ningún tipo de control central (en ese aspecto, es similar a Ethernet). El otro, llamado **PCF** (**Función de Coordinación Puntual),** utiliza la estación base para controlar toda la actividad en su celda. Todas las implementaciones soportan DCF pero PCF es opcional. A continuación analizaremos estos dos modos a la vez.

Cuando se emplea DCF, 802.11 utiliza un protocolo llamado **CSMA/CA** (**CSMA con Evitación de Colisiones).** En este protocolo, se utiliza tanto la detección del canal físico como la del canal virtual. Los dos métodos de funcionamiento son soportados por CSMA/CA. En el primer método, cuando una estación desea transmitir, detecta el canal. Si está inactivo, comienza a transmitir. No detecta el canal mientras transmite pero emite su trama completa, la cual podría ser destruida en el receptor debido a interferencia. Si el canal está ocupado, el emisor espera hasta que esté inactivo para comenzar a transmitir. Si ocurre una colisión, las estaciones involucradas en ella esperan un tiempo aleatorio, mediante el algoritmo de retroceso exponencial binario de Ethernet, y vuelve a intentarlo más tarde.

El otro modo de la operación CSMA/CA se basa en MACAW y utiliza la detección de canal virtual, como se ilustra en la figura 4-27. En este ejemplo, *A* desea enviar a *B*. *C* es una estación que está dentro del alcance de *A* (y posiblemente dentro del alcance de *B*, pero eso no importa). *D* es una estación dentro del alcance de *B* pero no dentro del de *A*.

El protocolo inicia cuando *A* decide enviar datos a *B*. *A* inicia enviándole una trama RTS a *B* en la que le solicita permiso para enviarle una trama. Cuando *B* recibe esta solicitud, podría decidir otorgarle el permiso, en cuyo caso le regresa una trama CTS. Al recibir la CTS, *A* ahora envía su

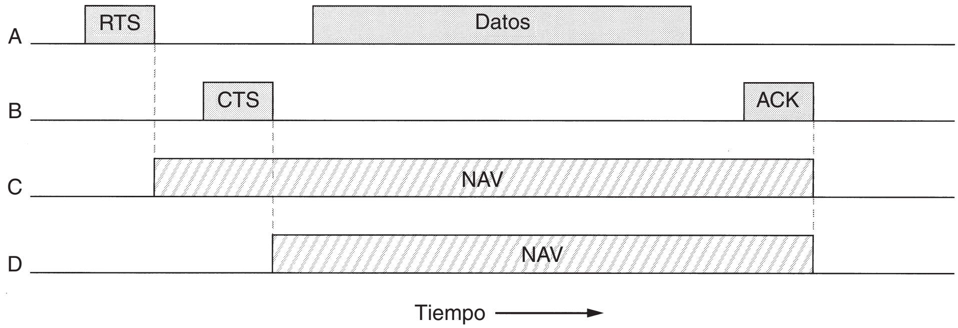


Figura 4-27. El uso de la detección de canal virtual utilizando CSMA/CA.

trama y comienza su temporizador de ACK. Al recibir correctamente la trama de datos, *B* responde con una trama de ACK, con lo que termina el intercambio. Si el temporizador de ACK de *A* termina antes de que el ACK regrese, todo el protocolo se ejecuta de nuevo.

Ahora consideremos este intercambio desde el punto de vista de *C* y *D*. *C* está dentro del alcance de *A*, por lo que podría recibir la trama RTS. Si pasa esto, se da cuenta de que alguien va a enviar datos pronto, así que por el bien de todos desiste de transmitir cualquier cosa hasta que el intercambio esté completo. A partir de la información proporcionada en la solicitud RTS, *C* puede estimar cuánto tardará la secuencia, incluyendo el ACK final, por lo que impone para sí misma un tipo de canal virtual ocupado, indicado por **NAV** (**Vector de Asignación de Red**) en la figura 4-27. *D* no escucha el RTS, pero sí el CTS, por lo que también impone la señal *NAV* para sí misma. Observe que las señales *NAV* no se transmiten; simplemente son recordatorios internos para mantenerse en silencio durante cierto periodo.

En contraste con las redes cableadas, las inalámbricas son ruidosas e inestables, en gran parte debido a los hornos de microondas, que también utilizan las bandas sin licencia ISM. Como consecuencia, la probabilidad de que una trama llegue a su destino se decrementa con la longitud de la trama. Si la probabilidad de que cualquier bit sea erróneo es p, entonces la probabilidad de que una trama de n bits se reciba por completo y correctamente es $(1 - p)^n$. Por ejemplo, para $p = 10^{-4}$, la probabilidad de recibir correctamente una trama Ethernet completa (12,144 bits) es menor que 30%. Si $p = 10^{-5}$, aproximadamente una trama de 9 estará dañada. Incluso si $p = 10^{-6}$, más de 1% de las tramas se dañará, lo que equivale a casi una docena por segundo, y más si se utilizan tramas más cortas que el máximo. En resumen, si una trama es demasiado grande, tiene muy pocas probabilidades de pasar sin daño y probablemente tenga que retransmitirse.

Para solucionar el problema de los canales ruidosos, 802.11 permite dividir las tramas en fragmentos, cada uno con su propia suma de verificación. Cada fragmento se numera de manera individual y su recepción se confirma utilizando un protocolo de parada y espera (es decir, el emisor podría no transmitir fragmentos de $k + 1$ hasta que haya recibido la confirmación de recepción del fragmento k). Una vez que se ha adquirido el canal mediante RTS y CTS, pueden enviarse múltiples

fragmentos en una fila, como se muestra en la figura 4-28. La secuencia de fragmentos se conoce como **ráfaga de fragmentos.**

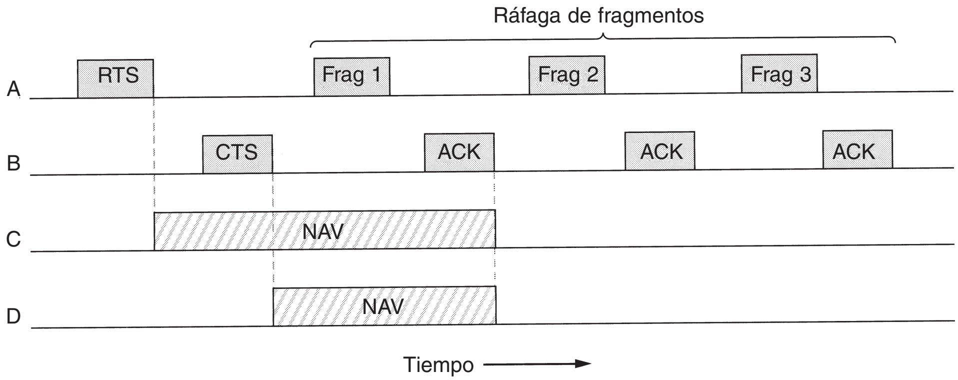


Figura 4-28. Una ráfaga de fragmentos.

La fragmentación incrementa la velocidad real de transporte restringiendo las retransmisiones a los fragmentos erróneos en lugar de la trama completa. El tamaño del fragmento no lo fija el estándar pero es un parámetro de cada celda y la estación base puede ajustarlo. El mecanismo NAV mantiene otras estaciones en silencio sólo hasta la siguiente confirmación de recepción, pero se utiliza otro mecanismo (descrito a continuación) para permitir que otra ráfaga de fragmentos completa se envíe sin interferencia.

Todo el análisis anterior se aplica al modo DCF 802.11. En él, no hay control central y la estación compite por tiempo aire, como en Ethernet. El otro modo permitido es PCF, en el que la estación base sondea las demás estaciones, preguntándoles si tienen tramas que enviar. Puesto que el orden de transmisión se controla por completo por la estación base en el modo PCF, no ocurren colisiones. El estándar prescribe el mecanismo para sondeo, pero no la frecuencia del sondeo, el orden del sondeo, ni el hecho de que las demás estaciones necesiten obtener un servicio igual.

El mecanismo básico consiste en que la estación base difunda una **trama de beacon** (trama guía o faro) de manera periódica (de 10 a 100 veces por segundo). Esta trama contiene parámetros de sistema, como secuencias de salto y tiempos de permanencia (para FHSS), sincronización de reloj, etcétera. También invita a las nuevas estaciones a suscribirse al servicio de sondeo. Una vez que una estación se inscribe para el servicio de sondeo a cierta tasa, se le garantiza de manera efectiva cierta fracción de ancho de banda, y se hace posible proporcionar garantías de calidad de servicio.

La duración de la batería siempre es un problema en los dispositivos inalámbricos móviles, por lo que 802.11 pone atención al asunto de la administración de energía. En particular, una estación base puede conducir una estación móvil al estado de hibernación hasta que dicha estación base o el usuario la saquen de él de manera explícita. Sin embargo, el hecho de indicar a una estación que entre en estado de hibernación significa que la estación base tiene la responsabilidad de almacenar en el búfer las tramas que vayan dirigidas a ella mientras la estación móvil esté hibernando. Posteriormente, esas tramas pueden colectarse.

PCF y DCF pueden coexistir dentro de una celda. Al principio podría parecer imposible tener control central y distribuido funcionando al mismo tiempo, pero 802.11 proporciona una forma de

alcanzar este objetivo. Funciona definiendo cuidadosamente el intervalo de tiempo entre tramas. Después de que se ha enviado una trama, se necesita cierta cantidad de tiempo muerto antes de que cualquier estación pueda enviar una trama. Se definen cuatro intervalos diferentes, cada uno con un propósito específico. Estos intervalos se describen en la figura 4-29.

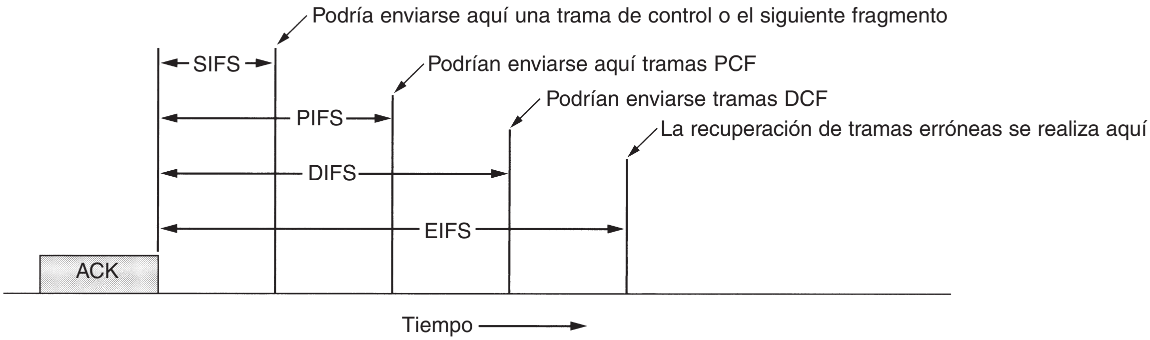


Figura 4-29. Espaciado entre tramas 802.11.

El intervalo más corto es **SIFS** (**Espaciado Corto Entre Tramas**). Se utiliza para permitir que las distintas partes de un diálogo transmitan primero. Esto incluye dejar que el receptor envíe un CTS para responder a una RTS, dejar que el receptor envíe un ACK para un fragmento o una trama con todos los datos y dejar que el emisor de una ráfaga de fragmentos transmita el siguiente fragmento sin tener que enviar una RTS nuevamente.

Siempre hay una sola estación que debe responder después de un intervalo SIFS. Si falla al utilizar su oportunidad y transcurre un tiempo **PIFS** (**Espaciado Entre Tramas PCF**), la estación base podría enviar una trama de *beacon* o una trama de sondeo. Este mecanismo permite que una estación base envíe una trama de datos o una secuencia de fragmentos para finalizar su trama sin que nadie interfiera, pero le da a la estación base la oportunidad de tomar el canal cuando el emisor anterior haya terminado, sin tener que competir con usuarios ansiosos.

Si la estación base no tiene nada que decir y transcurre un tiempo **DIFS** (**Espaciado Entre Tramas DCF**), cualquier estación podría intentar adquirir el canal para enviar una nueva trama. Se aplican las reglas de contención normales, y si ocurre una colisión, podría necesitarse el retroceso exponencial binario.

Sólo una estación que acaba de recibir una trama errónea o desconocida utiliza el último intervalo de tiempo, **EIFS** (**Espaciado Entre Tramas Extendido**), para reportar la trama errónea. La idea de dar a este evento la menor prioridad es que debido a que el receptor tal vez no tenga idea de lo que está pasando, debe esperar un tiempo considerable para evitar interferir con un diálogo en curso entre las dos estaciones.

4.4.4 La estructura de trama 802.11

El estándar 802.11 define tres clases diferentes de tramas en el cable: de datos, de control y de administración. Cada una de ellas tiene un encabezado con una variedad de campos utilizados

dentro de la subcapa MAC. Además, hay algunos encabezados utilizados por la capa física, pero éstos tienen que ver en su mayor parte con las técnicas de modulación utilizadas, por lo que no las trataremos aquí.

En la figura 4-30 se muestra el formato de la trama de datos. Primero está el campo de *Control de trama*. Éste tiene 11 subcampos. El primero es la *Versión de protocolo*, que permite que dos versiones del protocolo funcionen al mismo tiempo en la misma celda. Después están los campos de *Tipo* (de datos, de control o de administración) y de *Subtipo* (por ejemplo, RTS o CTS). Los bits *A DS* y *De DS* indican que la trama va hacia o viene del sistema de distribución entre celdas (por ejemplo, Ethernet). El bit *MF* indica que siguen más fragmentos. El bit *Retrans.* marca una retransmisión de una trama que se envió anteriormente. El bit de *Administración de energía* es utilizado por la estación base para poner al receptor en estado de hibernación o sacarlo de tal estado. El bit *Más* indica que el emisor tiene tramas adicionales para el receptor. El bit *W* especifica que el cuerpo de la trama se ha codificado utilizando el algoritmo **WEP** (**Privacidad Inalámbrica Equivalente**). Por último, el bit *O* indica al receptor que una secuencia de tramas que tenga este bit encendido debe procesarse en orden estricto.

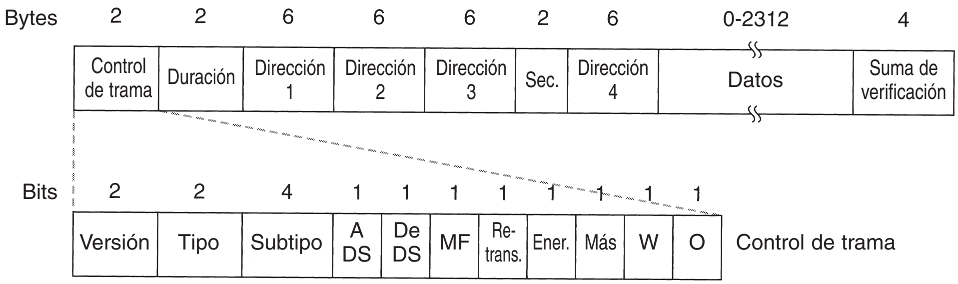


Figura 4-30. La trama de datos 802.11.

El segundo campo de la trama de datos, el de *Duración*, indica cuánto tiempo ocuparán el canal la trama y su confirmación de recepción. Este campo también está presente en las tramas de control y es la forma mediante la cual otras estaciones manejan el mecanismo NAV. El encabezado de trama contiene cuatro direcciones, todas en formato estándar IEEE 802. Obviamente se necesitan el origen y el destino, pero, ¿para qué son las otras dos? Recuerde que las tramas podrían entrar o dejar una celda a través de una estación base. Las otras dos direcciones se utilizan para las estaciones base de origen y destino para el tráfico entre celdas.

El campo de *Secuencia* permite que se numeren los fragmentos. De los 16 bits disponibles, 12 identifican la trama y 4 el fragmento. El campo de *Datos* contiene la carga útil, hasta 2312 bytes, y le sigue el campo común de *Suma de verificación*.

Las tramas de administración tienen un formato similar al de las tramas de datos, excepto que no tienen una de las direcciones de la estación base, debido a que las tramas de administración se restringen a una sola celda. Las tramas de control son más cortas; tienen una o dos direcciones, y no tienen ni campo de *Datos* ni de *Secuencia*. La información clave aquí se encuentra en el campo de *Subtipo*, que por lo general es RTS, CTS o ACK.

4.4.5 Servicios

El estándar 802.11 afirma que cada LAN inalámbrica que se apegue a él debe proporcionar nueve servicios. Éstos se dividen en dos categorías: cinco servicios de distribución y cuatro de estación. Los servicios de distribución se relacionan con la administración de membresías dentro de la celda y con la interacción con estaciones que están fuera de la celda. En contraste, los servicios de estación se relacionan con la actividad dentro de una sola celda.

Los cinco servicios de distribución son proporcionados por las estaciones base y tienen que ver con la movilidad de la estación conforme entran y salen de las celdas, conectándose ellos mismos a las estaciones base y separándose ellos mismos de dichas estaciones. Estos servicios son los siguientes:

1. **Asociación.** Este servicio es utilizado por las estaciones móviles para conectarse ellas mismas a las estaciones base. Por lo general, se utiliza después de que una estación se mueve dentro del alcance de radio de la estación base. Una vez que llega, anuncia su identidad y sus capacidades. Éstas incluyen las tasas de datos soportadas, necesarias para los servicios PCF (es decir, el sondeo), y los requerimientos de administración de energía. La estación base podría aceptar o rechazar la estación móvil. Si se acepta, dicha estación debe autenticarse.

2. **Disociación.** Es posible que la estación o la estación base se disocie, con lo que se rompería la relación. Una estación podría utilizar este servicio antes de apagarse o de salir, pero la estación base también podría utilizarlo antes de su mantenimiento.

3. **Reasociación.** Una estación podría cambiar su estación base preferida mediante este servicio. Esta capacidad es útil para estaciones móviles que se mueven de una celda a otra. Si se utiliza correctamente, no se perderán datos como consecuencia del cambio de estación base (*handover*). (Pero 802.11, al igual que Ethernet, es sólo un servicio de mejor esfuerzo.)

4. **Distribución.** Este servicio determina cómo enrutar tramas enviadas a la estación base. Si el destino es local para la estación base, las tramas pueden enviarse directamente a través del aire. De lo contrario, tendrán que reenviarse a través de la red cableada.

5. **Integración.** Si una trama necesita enviarse a través de una red no 802.11 con un esquema de direccionamiento o formato de trama diferentes, este servicio maneja la traducción del formato 802.11 al requerido por la red de destino.

Los cuatro servicios restantes son dentro de las celdas (es decir, se relacionan con acciones dentro de una sola celda). Se utilizan después de que ha ocurrido la asociación y son las siguientes:

1. **Autenticación.** Debido a que las estaciones no autorizadas pueden recibir o enviar con facilidad la comunicación inalámbrica, una estación debe autenticarse antes de que se le permita enviar datos. Una vez que la estación base asocia una estación móvil (es decir, la ha aceptado en su celda), le envía una trama especial de desafío para ver si dicha estación móvil sabe la clave secreta (contraseña) que se le ha asignado. La estación móvil prueba que sabe la clave secreta codificando la trama de desafío y regresándola a la estación base. Si el resultado es correcto, la estación móvil se vuelve miembro de la celda. En el estándar inicial, la estación base no tiene que probar su identidad a la estación móvil, pero se está realizando trabajo para reparar este defecto en el estándar.

2. **Desautenticación.** Cuando una estación previamente autenticada desea abandonar la red, se desautentica. Después de esto, tal vez ya no utilice la red.

3. **Privacidad.** Para que la información que se envía a través de una LAN inalámbrica se mantenga confidencial, debe codificarse. Este servicio maneja la codificación y la decodificación. El algoritmo de codificación especificado es RC4, inventado por Ronald Rivest del M.I.T.

4. **Entrega de datos.** Por último, la transmisión de datos es la parte esencial, por lo que el 802.11 naturalmente proporciona una forma de transmitir y recibir datos. Puesto que el 802.11 está basado en Ethernet y no se garantiza que la transmisión a través de Ethernet sea 100% confiable, tampoco se garantiza que la transmisión a través del 802.11 sea confiable. Las capas superiores deben tratar con la detección y la corrección de errores.

Una celda 802.11 tiene algunos parámetros que pueden inspeccionarse y, en algunos casos, ajustarse. Se relacionan con la codificación, intervalos de expiración de temporizador, tasas de datos, frecuencia de la trama de *beacon*, etcétera.

Las LANs inalámbricas basadas en 802.11 se están comenzando a distribuir en edificios de oficinas, aeropuertos, hoteles, restaurantes y universidades de todo el mundo. Se espera un crecimiento rápido. Para obtener información adicional acerca de la distribución extendida de 802.11 en CMU, vea (Hills, 2001).

4.5 BANDA ANCHA INALÁMBRICA

Hemos estado en casa mucho tiempo. Salgamos y veamos si hay algo interesante sobre redes por ahí. Pues sucede que hay bastantes novedades, y algunas de ellas tienen que ver con una característica llamada última milla. Con la desregulación del sistema telefónico en muchos países, en la actualidad a los competidores de la compañía telefónica arraigada con frecuencia se les permite ofrecer voz local y servicio de alta velocidad de Internet. Ciertamente hay mucha demanda. El problema es que el tendido de fibra óptica, cable coaxial o incluso cable de par

trenzado categoría 5 a millones de casas y oficinas es extremadamente costoso. ¿Qué debe hacer un competidor?

La respuesta es la banda ancha inalámbrica. Construir una antena enorme en una colina en las afueras del pueblo e instalar antenas que se dirijan a dicha antena en los techos de los clientes es más fácil y barato que cavar zanjas y ensartar cables. Por lo tanto, las compañías de telecomunicación en competencia tienen mucho interés en proporcionar un servicio de comunicación inalámbrica de multimegabits para voz, Internet, películas bajo demanda, etcétera. Como vimos en la figura 2-30, los LMDS se inventaron para este propósito. Sin embargo, hasta hace poco, cada portadora diseñaba su propio sistema. Esta falta de estándares significaba que el hardware y software no se podía producir en masa, por lo que los precios eran altos y la aceptación, baja.

Muchas personas en la industria se dieron cuenta de que tener un estándar de banda ancha inalámbrica era el elemento clave que faltaba, por lo que se le pidió a IEEE que formara un comité compuesto de personas de compañías clave y de academias para redactar el estándar. El siguiente número disponible en el espacio de numeración 802 era **802.16,** por lo que el estándar obtuvo este número. El trabajo se inició en julio de 1999, y el estándar final se aprobó en abril de 2002. Oficialmente el estándar se llama "Air Interface for Fixed Broadband Wireless Access Systems" (Interfaz de Aire para Sistemas Fijos de Acceso Inalámbrico de Banda Ancha). Sin embargo, algunas personas prefieren llamarlo **MAN (red de área metropolitana) inalámbrica** o **circuito local inalámbrico.** Nos referiremos a estos términos de manera indistinta.

Al igual que los otros estándares 802, el 802.16 estuvo influido fuertemente por el modelo OSI, incluyendo las (sub)capas, terminología, primitivas de servicios y más. Desgraciadamente, al igual que OSI, es muy complicado. En las siguientes secciones daremos una breve descripción de algunos de los puntos de importancia del estándar 802.16, pero este tratado no es completo y no trata muchos detalles. Para mayor información acerca de la banda ancha inalámbrica, vea (Bolcskei y cols., 2001, y Webb, 2001). Para mayor información sobre el estándar 802.16 en particular, vea (Eklund y cols., 2002).

4.5.1 Comparación entre los estándares 802.11 y 802.16

En este punto tal vez piense: ¿Por qué diseñar un nuevo estándar? ¿Por qué no simplemente utilizar 802.11? Hay algunas buenas razones para no utilizar 802.11, principalmente porque 802.11 y 802.16 resuelven diferentes problemas. Antes de introducirnos en la tecnología de 802.16, probablemente valga la pena dar algunos detalles de por qué es necesario un estándar completamente nuevo.

Los entornos en los que funcionan 802.11 y 802.16 son similares en algunas formas, principalmente en que fueron diseñados para proporcionar comunicaciones inalámbricas de alto ancho de banda. Pero también difieren en aspectos muy importantes. Para empezar, el protocolo 802.16 proporciona servicio a edificios, y éstos no son móviles. No migran de celda a celda con frecuencia. La mayor parte de 802.11 tiene que ver con la movilidad, y nada de eso es relevante aquí. Además, los edificios pueden tener más de una computadora en ellos, lo cual no ocurre cuando la estación final es una sola computadora portátil.

Debido a que los dueños de edificios por lo general están dispuestos a gastar mucho más dinero en artículos de comunicación que los dueños de computadoras portátiles, hay mejores radios disponibles. Esta diferencia significa que 802.16 puede utilizar comunicación de dúplex total, algo que 802.11 evita para mantener bajo el costo de los radios.

Puesto que el estándar 802.16 se usa en parte de la ciudad, las distancias involucradas pueden ser de varios kilómetros, lo que significa que la energía detectada en la estación base puede variar considerablemente de estación en estación. Esta variación afecta la relación señal a ruido, que, a su vez, fija múltiples esquemas de modulación. Además, la comunicación abierta a través de la ciudad significa que la seguridad y privacidad son esenciales y obligatorias.

Además, es probable que cada celda tenga muchos más usuarios que una celda 802.11 típica, y se espera que estos usuarios utilicen más ancho de banda que un usuario 802.11 típico. Después de todo es raro que una compañía reúna a 50 empleados con sus computadoras portátiles en un cuarto para ver si pueden saturar la red inalámbrica 802.11 al observar a la vez 50 películas por separado. Por esta razón es necesario más espectro del que las bandas ISM pueden proporcionar, con lo que se obliga al estándar 802.16 a funcionar en el rango de frecuencia más alto de 10 a 66 GHz, el único lugar en el que el espectro no utilizado aún está disponible.

Pero estas ondas milimétricas tienen propiedades físicas diferentes que las ondas más largas en las bandas ISM, que a su vez requieren una capa física completamente diferente. Una propiedad de las ondas milimétricas es que el agua (especialmente la lluvia y, en cierta medida, la nieve, el granizo y, con un poco de mala suerte, la niebla espesa) las absorbe por completo. En consecuencia, el control de errores es más importante que en un entorno interno. Las ondas milimétricas pueden enfocarse en rayos direccionales (802.11 es omnidireccional), por lo que las opciones realizadas en 802.11 relacionadas con la propagación de múltiples rutas son debatibles.

Otro aspecto es la calidad del servicio. Si bien el estándar 802.11 proporciona soporte para el tráfico en tiempo real (utilizando el modo PCF), realmente no se diseñó para uso extenso de telefonía y multimedia. En contraste, se espera que el estándar 802.16 soporte estas aplicaciones por completo porque está diseñado para uso residencial y de negocios.

En resumen, 802.11 se diseñó para ser una Ethernet móvil, mientras que el estándar 802.16 se diseñó para ser televisión por cable inalámbrica, pero estacionaria. Estas diferencias son tan grandes que los estándares resultantes son muy diferentes debido a que tratan de optimizar cosas distintas.

Vale la pena hacer una pequeña comparación con el sistema de teléfonos celulares. Al referirnos a los teléfonos celulares, hablamos de estaciones móviles de banda estrecha, baja potencia y con orientación a voz que se comunican mediante microondas de longitud media. Nadie ve (todavía) películas de 2 horas a alta resolución en teléfonos móviles GSM. Incluso UMTS tiene poca esperanza de cambiar esta situación. En resumen, el mundo de las MANs inalámbricas es más demandante que el de los teléfonos móviles, por lo que se necesita un sistema completamente diferente. El hecho de que en el futuro el estándar 802.16 pueda utilizarse para dispositivos móviles es una pregunta interesante. No fue optimizado para ellos, pero la posibilidad está abierta. Por el momento se enfoca en los sistemas inalámbricos fijos.

4.5.2 La pila de protocolos del estándar 802.16

En la figura 4-31 se ilustra la pila de protocolos del estándar 802.16. La estructura general es similar a la de las otras redes 802, pero con más subcapas. La subcapa inferior tiene que ver con la transmisión. El radio de banda estrecha tradicional se utiliza con esquemas de modulación tradicionales. Arriba de la capa de transmisión física está una subcapa de convergencia para ocultarle las diferentes tecnologías a la capa de enlace de datos. En la actualidad el estándar 802.11 también tiene algo parecido a esto, sólo que el comité eligió no formalizarlo con un nombre de tipo OSI.

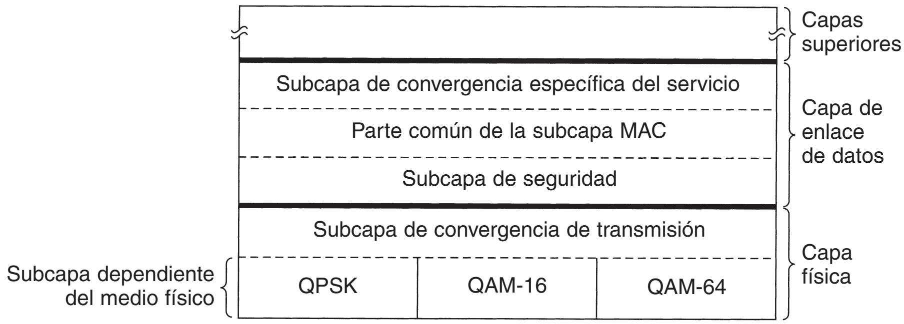


Figura 4-31. La pila de protocolos del 802.16.

Aunque no los mostramos en la figura, se está trabajando para agregar dos nuevos protocolos de capa física. El estándar 802.16a soportará a OFDM en el rango de frecuencia de 2 a 11 GHz. El estándar 802.16b operará en la banda ISM de 5 GHz. Estos dos son intentos para acercarse al estándar 802.11.

La capa de enlace de datos consta de tres subcapas. La inferior tiene que ver con la privacidad y seguridad, lo cual es más importante para las redes externas públicas que para las redes internas privadas. Maneja codificación, decodificación y administración de claves.

A continuación se encuentra la parte común de la subcapa MAC. Es aquí donde se encuentran los principales protocolos, como la administración de canal. El modelo consiste en que la estación base controla el sistema. Puede calendarizar de manera muy eficiente los canales de flujo descendente (es decir, de la estación base al suscriptor) y es muy importante en el manejo de los canales ascendentes (es decir, del suscriptor a la estación base). Una característica no muy común de la subcapa MAC es que, a diferencia de las subcapas de las otras redes 802, es completamente orientada a la conexión, para proporcionar garantías de calidad del servicio para la comunicación de telefonía y multimedia.

En los otros protocolos 802, la subcapa de convergencia específica del servicio toma el lugar de la subcapa de enlace lógico. Su función es interactuar con la capa de red. Un problema aquí es que el estándar 802.16 fue diseñado para integrarse sin ningún problema tanto con los protocolos

de datagramas (por ejemplo, PPP, IP y Ethernet) como con ATM. El problema es que los protocolos de paquetes no son orientados a la conexión y ATM sí lo es. Esto significa que cada conexión ATM se tiene que asignar a una conexión 802.16, que en principio es un asunto directo. ¿Pero en cuál conexión 802.16 debe asignarse un paquete IP entrante? El problema se soluciona en esta subcapa.

4.5.3 La capa física del estándar 802.16

Como se mencionó anteriormente, la banda ancha inalámbrica necesita mucho espectro y el único lugar para encontrarlo es en el rango de 10 a 66 GHz. Estas ondas milimétricas tienen una propiedad interesante que las microondas más largas no tienen: viajan en líneas rectas, a diferencia del sonido, pero en forma similar a la luz. Como consecuencia, la estación base puede tener múltiples antenas, cada una apuntando a un sector diferente del terreno circundante, como se muestra en la figura 4-32. Cada sector tiene sus propios usuarios y es completamente independiente de los sectores adyacentes, algo que no es verdad es el radio celular, el cual es omnidireccional.

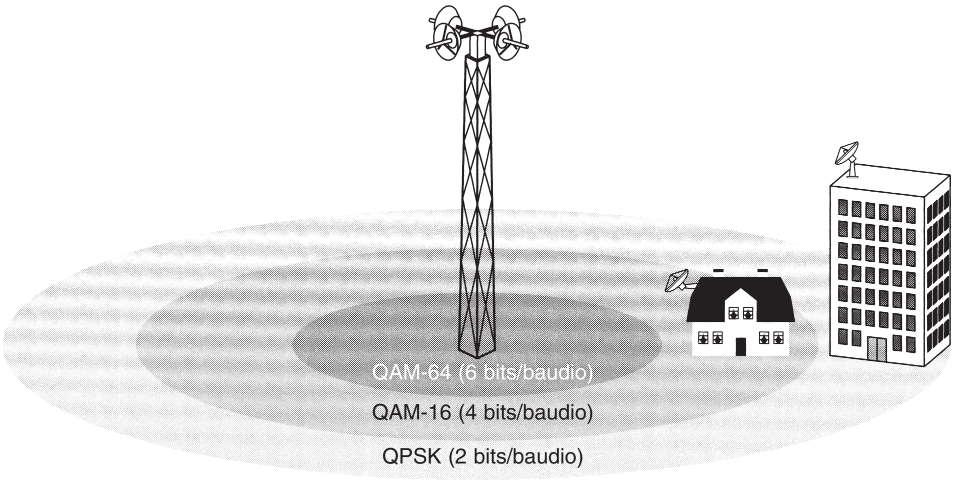


Figura 4-32. El entorno de transmisión 802.16.

Debido a que la fuerza de señal en la banda milimétrica desciende drásticamente con la distancia a partir de la estación base, la relación señal a ruido también desciende con la distancia a partir de la estación base. Por esta razón, el estándar 802.16 emplea tres esquemas de modulación diferentes, dependiendo de la distancia entre la estación suscriptora y la estación base. Para suscriptores cercanos se utiliza QAM-64, con 6 bits/baudio. Para suscriptores a distancias medias se utiliza QAM-16, con 4 bits/baudio. Para suscriptores distantes se utiliza QPSK, con 2 bits/baudio. Por ejemplo, para un valor típico de 25 MHz digno de espectro, QAM-64 da 150 Mbps, QAM-16 da 100 Mbps y QPSK da 50 Mbps. En otras palabras, entre más lejos esté el suscriptor de la estación base, será más baja la tasa de datos (algo similar a lo que vimos con ADSL en la figura 2-27). El diagrama de constelación de estas tres técnicas de modulación se mostró en la figura 2-25.

Dado el objetivo de producir un sistema de banda ancha, sujeto a las limitantes físicas mostradas anteriormente, los diseñadores del protocolo 802.16 trabajaron duro para utilizar eficientemente el espectro disponible. Los que no les gustaba era la forma en que funcionaban GSM y DAMPS. Ambos utilizan bandas de frecuencia diferentes pero iguales para el tráfico ascendente y descendente. Para voz, es probable que el tráfico sea simétrico en su mayor parte, pero para acceso a Internet por lo general hay más tráfico descendente que ascendente. En consecuencia, el estándar 802.16 proporciona una forma más flexible para asignar el ancho de banda. Se utilizan dos esquemas: **FDD** (**Duplexación por División de Frecuencia**) y **TDD** (**Duplexación por División de Tiempo**)**.** Este último se ilustra en la figura 4-33. Aquí la estación base envía tramas periódicamente. Cada trama contiene ranuras de tiempo. Las primeras son para el tráfico descendente. Después se encuentra el tiempo de protección o guarda, el cual es utilizado por las estaciones para cambiar de dirección. Por último, están las ranuras para el tráfico ascendente. El número de ranuras de tiempo dedicadas para cada dirección se puede cambiar de manera dinámica con el fin de que el ancho de banda en cada dirección coincida con el tráfico.

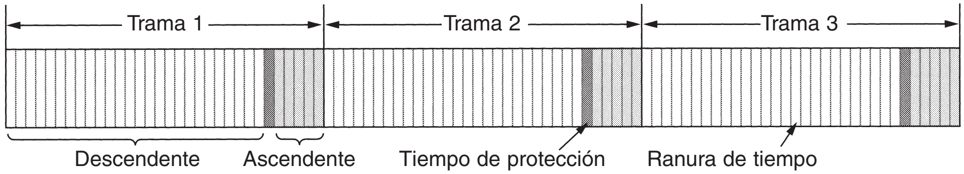


Figura 4-33. Tramas y ranuras de tiempo para duplexación por división de tiempo.

La estación base asigna el tráfico descendente en ranuras de tiempo. Además, controla por completo esta dirección. El tráfico ascendente es más complejo y depende de la calidad del servicio requerido. Más adelante volveremos a la asignación de ranuras, cuando analicemos la subcapa MAC.

Otra característica interesante de la capa física es su capacidad de empaquetar múltiples tramas MAC consecutivas en una sola transmisión física. Esta característica mejora la eficiencia espectral al reducir el número de preámbulos y encabezados de capa física necesarios.

Otro aspecto que vale la pena mencionar es el uso de los códigos de Hamming para realizar corrección de errores hacia delante en la capa física. La mayoría de las otras redes se basa simplemente en sumas de verificación para detectar errores y solicitar retransmisiones cuando se reciben tramas erróneas. Pero en el entorno de banda ancha de área amplia, se esperan tantos errores de transmisión que la corrección de errores se emplea en la capa física, además de sumas de verificación en las capas superiores. El objetivo de la corrección de errores es hacer que el canal luzca mejor de lo que realmente es (de la misma forma en que los CD-ROMs parecen ser muy confiables, pero sólo porque más de la mitad de los bits se destinan para la corrección de errores en la capa física).

4.5.4 El protocolo de la subcapa MAC del 802.16

La capa de enlace de datos se divide en tres subcapas, como vimos en la figura 4-31. Debido a que estudiaremos la criptografía hasta el capítulo 8 es difícil explicar ahora cómo funciona la

subcapa de seguridad. Basta decir que la codificación se utiliza para que todos los datos transmitidos se mantengan en secreto. Sólo las cargas útiles de las tramas se codifican; los encabezados no se codifican. Esta propiedad significa que un fisgón puede ver quién está hablándole a quién pero no puede saber qué se están diciendo.

Si usted ya sabe algo sobre criptografía, a continuación se muestra una explicación de un párrafo acerca de la subcapa de seguridad. Si no sabe nada sobre criptografía, es probable que el siguiente párrafo no sea muy ilustrativo para usted (pero podría considerar volver a leerlo después de terminar el capítulo 8).

Cuando un suscriptor se conecta a una estación base, realiza autenticación mutua con criptografía de clave pública RSA mediante certificados X.509. Las cargas útiles mismas se codifican mediante un sistema de clave simétrica, ya sea DES con cambio de bloques de código o triple DES con dos claves. Es probable que AES (Rijndael) se agregue pronto. La verificación de integridad utiliza SHA-1. Eso no es tan malo, ¿o sí?

Ahora veamos la parte común de la subcapa MAC. Las tramas MAC ocupan un número integral de ranuras de tiempo de la capa física. Cada trama se compone de subtramas, de las cuales las primeras dos son los mapas descendente y ascendente. Éstos indican lo que hay en cada ranura de tiempo y cuáles ranuras de tiempo están libres. El mapa descendente también contiene varios parámetros de sistema para informar de nuevas estaciones conforme entran en línea.

El canal descendente es muy directo. La estación base decide simplemente lo que se va a poner en cada subtrama. El canal ascendente es más complicado debido a que hay suscriptores no coordinados compitiendo por él. Su asignación está estrechamente relacionada con el aspecto de calidad del servicio. Hay cuatro clases de servicio definidas:

1. Servicio de tasa de bits constante.
2. Servicio de tasa de bits variable en tiempo real.
3. Servicio de tasa de bits variable no en tiempo real.
4. Servicio de mejor esfuerzo.

Todos los servicios del estándar 802.16 son orientados a la conexión, y cada conexión toma una de las clases de servicio mostradas anteriormente, que se determina cuando se configura la conexión. Este diseño es muy diferente al de 802.11 o al de Ethernet, los cuales no tienen conexiones en la subcapa MAC.

El servicio de tasa de bits constante está diseñado para transmitir voz descomprimida, como en un canal T1. Este servicio necesita enviar una cantidad predeterminada de datos en intervalos de tiempo predeterminados. Se aloja mediante la dedicación de ciertas ranuras de tiempo a cada conexión de este tipo. Una vez que se ha asignado el ancho de banda, las ranuras de tiempo quedan disponibles automáticamente, sin necesidad de solicitar cada una.

El servicio de tasa de bits variable en tiempo real está destinado para la multimedia comprimida y otras aplicaciones en tiempo real en las que la cantidad de ancho de banda necesaria puede variar en cada instante. Es ajustada por la estación base sondeando al suscriptor a un intervalo fijo para saber cuánto ancho de banda se necesita esta vez.

El servicio de tasa de bits variable en tiempo no real es para las transmisiones pesadas que no son en tiempo real, como transmisiones grandes de archivos. Para este servicio, la estación base sondea al suscriptor con mucha frecuencia. Un cliente de tasa de bits constante puede establecer un bit en una de sus tramas, solicitando un sondeo para enviar tráfico adicional (tasa de bits variable).

Si una estación no responde a un sondeo *k* veces en una fila, la estación base la coloca en un grupo de multidifusión y elimina su sondeo personal. En su lugar, cuando se sondea el grupo de multidifusión, cualquiera de las estaciones que conforman el grupo puede responder, compitiendo por el servicio. De esta forma, las estaciones con poco tráfico no desperdician sondeos valiosos.

Por último, el servicio de mejor esfuerzo es para todo lo demás. No se realiza sondeo y el suscriptor debe competir por ancho de banda con otros suscriptores de mejor servicio. Las solicitudes por ancho de banda se realizan en ranuras de tiempo que están marcadas en el mapa ascendente como disponibles para competencia. Si una solicitud es exitosa, su éxito se notará en el siguiente mapa de bits descendente. Si no es exitosa, los suscriptores no exitosos deberán tratar más tarde. Para minimizar las colisiones, se utiliza el algoritmo de retroceso exponencial binario.

El estándar define dos formas de asignación de ancho de banda: por estación y por conexión. En el primer caso, la estación suscriptora agrega las necesidades de todos los usuarios del edificio y realiza solicitudes colectivas por ellos. Cuando se le concede el ancho de banda, lo asigna entre sus usuarios como considere necesario. En el último caso, la estación base administra cada conexión de manera directa.

4.5.5 La estructura de trama 802.16

Todas las tramas MAC comienzan con un encabezado genérico. A éste le sigue una carga útil y una suma de verificación (CRC) opcionales, como se ilustra en la figura 4-34. La carga útil no es necesaria en las tramas de control, por ejemplo, en las que solicitan ranuras de canal. La suma de verificación también es (sorprendentemente) opcional, debido a la corrección de errores en la capa física y al hecho de que nunca se realiza un intento por retransmitir tramas en tiempo real. Si estos intentos nunca ocurren, ¿para qué molestarse con una suma de verificación?

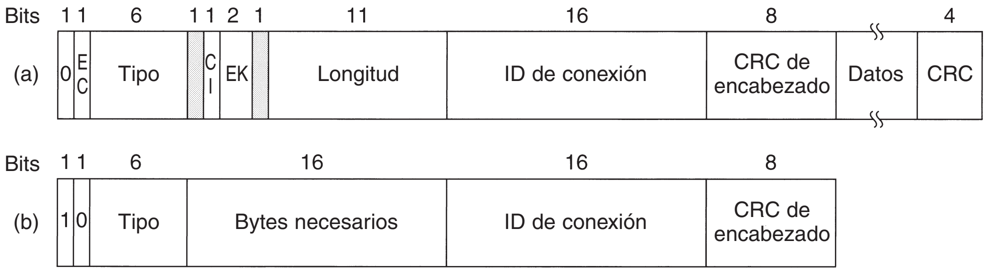


Figura 4-34. (a) Una trama genérica. (b) Una trama de solicitud de ancho de banda.

Un rápido vistazo a los campos de encabezado de la figura 4-34(a) es como sigue: el bit *EC* indica si la carga útil está encriptada. El campo *Tipo* identifica el tipo de la trama e indica principalmente si hay empaquetamiento y fragmentación. El campo *CI* indica la presencia o ausencia de la suma de verificación final. El campo *EK* indica cuál de las claves de encriptación se está utilizando (si es que se está utilizando alguna). El campo *Longitud* proporciona la longitud exacta de la trama, incluyendo la del encabezado. El *Identificador de conexión* indica a cuál conexión pertenece esta trama. Por último, el campo *CRC de encabezado* es la suma de verificación sólo del encabezado, que utiliza el polinomio $x^8 + x^2 + x + 1$.

En la figura 4-34(b) se muestra un segundo tipo de encabezado, para tramas que solicitan ancho de banda. Comienza con un bit 1 en lugar de uno 0 y es similar al encabezado genérico, excepto que el segundo y tercer bytes forman un número de 16 bits, lo que indica la cantidad de ancho de banda necesaria para transmitir el número de bytes especificados. Las tramas de solicitud de ancho de banda no transmiten datos útiles o un CRC de la trama completa.

Es posible decir muchísimas cosas más sobre el estándar 802.16, pero este no es el lugar para decirlo. Para mayor información, consulte el estándar mismo.

4.6 BLUETOOTH

En 1994, la empresa L. M. Ericsson se interesó en conectar sus teléfonos móviles y otros dispositivos (por ejemplo, PDAs,) sin necesidad de cables. En conjunto con otras cuatro empresas (IBM, Intel, Nokia y Toshiba), formó un SIG (grupo de interés especial, es decir, un consorcio) con el propósito de desarrollar un estándar inalámbrico para interconectar computadoras, dispositivos de comunicaciones y accesorios a través de radios inalámbricos de bajo consumo de energía, corto alcance y económicos. Al proyecto se le asignó el nombre **Bluetooth,** en honor de Harald Blaatand (Bluetooth) II (940-981), un rey vikingo que unificó (es decir, conquistó) Dinamarca y Noruega, también sin necesidad de cables. Aunque la idea original eran tan sólo prescindir de cables entre dispositivos, su alcance se expandió rápidamente al área de las LANs inalámbricas. Aunque esta expansión le dio más utilidad al estándar, también provocó el surgimiento de competencia con el 802.11. Para empeorar las cosas, los dos sistemas interfieren entre sí en el ámbito eléctrico. Asimismo, vale la pena hacer notar que Hewlett-Packard introdujo hace algunos años una red infrarroja para conectar periféricos de computadora sin cables, pero en realidad nunca alcanzó popularidad.

Sin desanimarse por esto, el SIG de Bluetooth emitió en julio de 1999 una especificación de 1500 páginas acerca de V1.0. Un poco después, el grupo de estándares del IEEE que se encarga de las redes de área personal inalámbricas, 802.15, adoptó como base el documento sobre Bluetooth y empezó a trabajar en él. A pesar de que podría parecer extraño estandarizar algo que ya cuenta con una especificación bien detallada, sin implementaciones incompatibles que tengan que armonizarse, la historia demuestra que al existir un estándar abierto manejado por un cuerpo neutral como el IEEE con frecuencia se estimula el uso de una tecnología. Para ser un poco más precisos, debe apuntarse que la especificación de Bluetooth está dirigida a un sistema completo, de la capa física a la capa de aplicación. El comité 802.15 del IEEE estandariza solamente las capas física y la de enlace de datos; el resto de la pila de protocolos está fuera de sus estatutos.

Aun cuando el IEEE aprobó en el 2002 el primer estándar para redes de área personal, 802.15.1, el SIG de Bluetooth continúa las mejoras. A pesar de que las versiones del SIG y del IEEE difieren, se espera que en breve coincidirán en un solo estándar.

4.6.1 Arquitectura de Bluetooth

Empecemos nuestro análisis del sistema Bluetooth con un rápido vistazo de sus elementos y de su propósito. La unidad básica de un sistema Bluetooth es una **piconet,** que consta de un nodo maestro y hasta siete nodos esclavos activos a una distancia de 10 metros. En una misma sala (grande) pueden encontrarse varias *piconets* y se pueden conectar mediante un nodo puente, como se muestra en la figura 4-35. Un conjunto de *piconets* interconectadas se denomina **scatternet.**

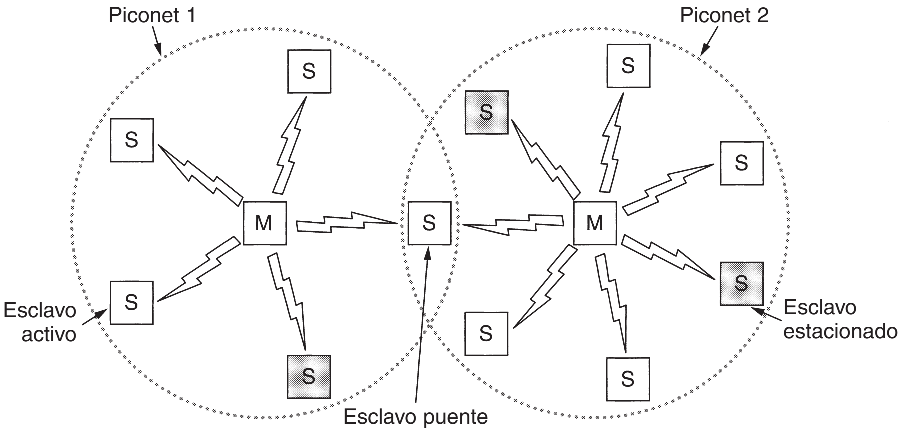


Figura 4-35. Dos piconets se pueden conectar para conformar una scatternet.

Además de los siete nodos esclavos activos de una *piconet*, puede haber hasta 255 nodos estacionados en la red. Éstos son dispositivos que el nodo maestro ha cambiado a un estado de bajo consumo de energía para reducir el desgaste innecesario de sus pilas. Lo único que un dispositivo en estado estacionado puede hacer es responder a una señal de activación por parte del maestro. También hay dos estados intermedios, *hold* y *sniff*, pero no nos ocuparemos de ellos aquí.

La razón para el diseño maestro/esclavo es que los diseñadores pretendían facilitar la implementación de chips Bluetooth completos por debajo de 5 dólares. La consecuencia de esta decisión es que los esclavos son sumamente pasivos y realizan todo lo que los maestros les indican. En esencia, una *piconet* es un sistema TDM centralizado, en el cual el maestro controla el reloj y determina qué dispositivo se comunica en un momento determinado. Todas las comunicaciones se realizan entre el maestro y el esclavo; no existe comunicación directa de esclavo a esclavo.

4.6.2 Aplicaciones de Bluetooth

La mayoría de los protocolos de red sólo proporcionan canales entre las entidades que se comunican y permiten a los diseñadores de aplicaciones averiguar para qué desean utilizarlos. Por ejemplo, el 802.11 no especifica si los usuarios deben utilizar sus computadoras portátiles para leer correo electrónico, navegar por la Red o cualquier otro uso. En contraste, la especificación Bluetooth V1.1 designa el soporte de 13 aplicaciones en particular y proporciona diferentes pilas de protocolos para cada una. Desgraciadamente, este enfoque conlleva una gran complejidad, que aquí omitiremos. En la figura 4-36 se describen las 13 aplicaciones, las cuales se denominan **perfiles.** Al analizarlas brevemente en este momento, veremos con mayor claridad lo que pretende el SIG de Bluetooth.

Nombre	Descripción
Acceso genérico	Procedimientos para el manejo de enlaces
Descubrimiento de servicios	Protocolo para descubrir los servicios que se ofrecen
Puerto serie	Reemplazo para un cable de puerto serie
Intercambio genérico de objetos	Define la relación cliente-servidor para el traslado de objetos
Acceso a LAN	Protocolo entre una computadora móvil y una LAN fija
Acceso telefónico a redes	Permite que una computadora portátil realice una llamada por medio de un teléfono móvil
Fax	Permite que un fax móvil se comunique con un teléfono móvil
Telefonía inalámbrica	Conecta un handset (teléfono) con su estación base local
Intercom (Intercomunicador)	Walkie-talkie digital
Headset (Diadema telefónica)	Posibilita la comunicación de voz sin utilizar las manos
Envío de objetos	Ofrece una manera de intercambiar objetos simples
Transferencia de archivos	Proporciona una característica para transferencia de archivos más general
Sincronización	Permite a un PDA sincronizarse con otra computadora

Figura 4-36. Los perfiles de Bluetooth.

El perfil de acceso genérico no es realmente una aplicación, sino más bien la base sobre la cual se construyen las aplicaciones; su tarea principal es ofrecer una manera para establecer y mantener enlaces (canales) seguros entre el maestro y los esclavos. El perfil de descubrimiento de servicios también es relativamente genérico; los dispositivos lo utilizan para descubrir qué servicios ofrecen otros dispositivos. Se espera que todos los dispositivos Bluetooth implementen estos dos perfiles. Los restantes son opcionales.

El perfil de puerto serie es un protocolo de transporte que la mayoría de los perfiles restantes utiliza. Emula una línea serie y es especialmente útil para aplicaciones heredadas que requieren una línea serie.

El perfil de intercambio genérico define una relación cliente-servidor para el traslado de datos. Los clientes inician operaciones, pero tanto un cliente como un servidor pueden fungir como esclavo. Al igual que el perfil de puerto serie, es la base para otros perfiles.

El siguiente grupo de tres perfiles está destinado a la conectividad. El perfil de acceso a LAN permite a un dispositivo Bluetooth conectarse a una red fija; este perfil es competencia directa del estándar 802.11. El perfil de acceso telefónico a redes fue el propósito original de todo el proyecto; permite a una computadora portátil conectarse a un teléfono móvil que contenga un módem integrado, sin necesidad de cables. El perfil de fax es parecido al de acceso telefónico a redes, excepto que posibilita a máquinas de fax inalámbricas enviar y recibir faxes a través de teléfonos móviles sin que exista una conexión por cable entre ambos.

Los tres perfiles siguientes son para telefonía. El perfil de telefonía inalámbrica proporciona una manera de conectar el *handset* (teléfono) de un teléfono inalámbrico a la estación base. En la actualidad, la mayoría de los teléfonos inalámbricos no se puede utilizar también como teléfonos móviles, pero quizá en el futuro se puedan combinar los teléfonos inalámbricos y los móviles. El perfil intercom hace posible que dos teléfonos se conecten como *walkie-talkies*. Por último, con el perfil *headset* (diadema telefónica) se puede realizar comunicación de voz entre la diadema telefónica y su estación base, por ejemplo, para comunicarse telefónicamente sin necesidad de utilizar las manos al manejar un automóvil.

Los tres perfiles restantes sirven para intercambiar objetos entre dos dispositivos inalámbricos, como tarjetas de presentación, imágenes o archivos de datos. En particular, el propósito del perfil de sincronización es cargar datos en un PDA o en una computadora portátil cuando se está fuera de casa y de recabar estos datos al llegar a casa.

¿Era realmente necesario explicar en detalle todas estas aplicaciones y proporcionar diferentes pilas de protocolos para cada una? Tal vez no, pero esto se debió a que fueron diversos grupos de trabajo los que diseñaron las diferentes partes del estándar y cada uno se enfocó en su problema específico y generó su propio perfil. La ley de Conway podría aplicarse en esta situación. (En el número de abril de 1968 de la revista *Datamation,* Melvin Conway apuntó que si se asignan *n* personas a escribir un compilador, se obtendrá un compilador de *n* pasos, o, en forma más general, la estructura del software reflejará la estructura del grupo que la produjo.) Quizá dos pilas de protocolos habrían sido suficientes en lugar de 13, una para la transferencia de archivos y otra para posibilitar el flujo continuo de la comunicación en tiempo real.

4.6.3 La pila de protocolos de Bluetooth

El estándar Bluetooth cuenta con muchos protocolos agrupados con poco orden en capas. La estructura de capas no sigue el modelo OSI, el modelo TCP/IP, el modelo 802 o algún otro modelo conocido. Sin embargo, el IEEE se encuentra modificando actualmente Bluetooth para ajustarlo al modelo 802. En la figura 4-37 se muestra la arquitectura básica de protocolos de Bluetooth tal como la modificó el comité 802.

La capa inferior es la capa de radio física, la cual es bastante similar a la capa física de los modelos OSI y 802. Se ocupa de la transmisión y la modulación de radio. Aquí, gran parte del interés se enfoca en el objetivo de lograr que el sistema tenga un costo bajo para que pueda entrar al mercado masivo.

La capa de banda base tiene algunos puntos en común con la subcapa MAC, aunque también incluye elementos de la capa física. Se encarga de la manera en que el maestro controla las ranuras de tiempo y de que éstas se agrupen en tramas.

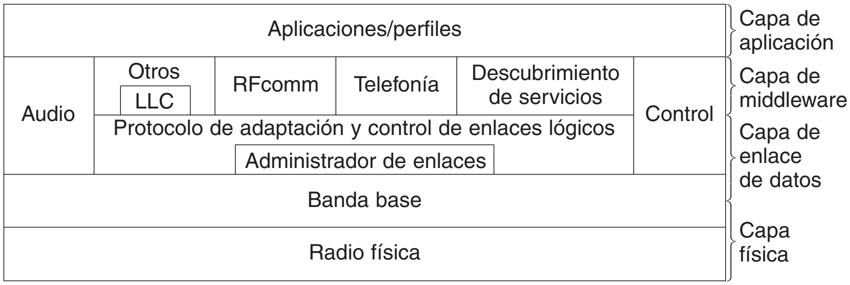


Figura 4-37. Versión 802.15 de la arquitectura de protocolos de Bluetooth.

A continuación se encuentra una capa con un grupo de protocolos un tanto relacionados. El administrador de enlaces se encarga de establecer canales lógicos entre dispositivos, incluyendo administración de energía, autenticación y calidad de servicio. El protocolo de adaptación y control de enlaces lógicos (también conocido como L2CAP) aísla a las capas superiores de los detalles de la transmisión. Es análogo a la subcapa LLC del estándar 802, pero difiere de ésta en el aspecto técnico. Como su nombre indica, los protocolos de audio y control se encargan del audio y el control, respectivamente. Las aplicaciones pueden acceder a ellos de manera directa sin necesidad de pasar por el protocolo L2CAP.

La siguiente capa hacia arriba es la de middleware, que contiene una mezcla de diferentes protocolos. El IEEE incorporó aquí la subcapa LLC del 802 para ofrecer compatibilidad con las redes 802. Los protocolos RFcomm, de telefonía y de descubrimiento de servicios son nativos. RFcomm (comunicación de Radio Frecuencia) es el protocolo que emula el puerto serie estándar de las PCs para la conexión de teclados, ratones y módems, entre otros dispositivos. Su propósito es permitir que dispositivos heredados lo utilicen con facilidad. El protocolo de telefonía es de tiempo real y está destinado a los tres perfiles orientados a voz. También se encarga del establecimiento y terminación de llamadas. Por último, el protocolo de descubrimiento de servicios se emplea para localizar servicios dentro de la red.

En la capa superior es donde se ubican las aplicaciones y los perfiles, que utilizan a los protocolos de las capas inferiores para realizar su trabajo. Cada aplicación tiene su propio subconjunto dedicado de protocolos. Por lo general, los dispositivos específicos, como las diademas telefónicas, contienen únicamente los protocolos necesarios para su aplicación.

En las siguientes secciones examinaremos las tres capas inferiores de la pila de protocolos de Bluetooth, ya que éstas corresponden más o menos con las subcapas física y MAC.

4.6.4 La capa de radio de Bluetooth

La capa de radio traslada los bits del maestro al esclavo, o viceversa. Es un sistema de baja potencia con un rango de 10 metros que opera en la banda ISM de 2.4 GHz. La banda se divide en 79 canales de 1 MHz cada uno. La modulación es por desplazamiento de frecuencia, con 1 bit por Hz, lo cual da una tasa de datos aproximada de 1 Mbps, pero gran parte de este espectro la consume la

sobrecarga. Para asignar los canales de manera equitativa, el espectro de saltos de frecuencia se utiliza a 1600 saltos por segundo y un tiempo de permanencia de 625 μseg. Todos los nodos de una *piconet* saltan de manera simultánea, y el maestro establece la secuencia de salto.

Debido a que tanto el 802.11 como Bluetooth operan en la banda ISM de 2.4 GHz en los mismos 79 canales, interfieren entre sí. Puesto que Bluetooth salta mucho más rápido que el 802.11, es más probable que un dispositivo Bluetooth dañe las transmisiones del 802.11 que en el caso contrario. Como el 802.11 y el 802.15 son estándares del IEEE, éste busca una solución para el problema, aunque no es tan sencilla porque ambos sistemas utilizan la banda ISM por la misma razón: no se requiere licencia para su uso. El estándar 802.11a emplea la otra banda ISM (5 GHz), pero tal estándar tiene un rango mucho más corto que el 802.11b (debido a la física de las ondas de radio), por lo cual el 802.11a no es una solución idónea en todos los casos. Algunas empresas han recurrido a la prohibición total de Bluetooth para solucionar el problema. Una solución de mercado es que la red más potente (en los aspectos político y económico, no eléctrico) solicite que la parte más débil modifique su estándar para dejar de interferir con ella. (Lansford y cols., 2001) dan algunas opiniones al respecto.

4.6.5 La capa de banda base de Bluetooth

La capa de banda base de Bluetooth es lo más parecido a una subcapa MAC. Esta capa convierte el flujo de bits puros en tramas y define algunos formatos clave. En la forma más sencilla, el maestro de cada *piconet* define una serie de ranuras de tiempo de 625 μseg y las transmisiones del maestro empiezan en las ranuras pares, y las de los esclavos, en las ranuras impares. Ésta es la tradicional multiplexión por división de tiempo, en la cual el maestro acapara la mitad de las ranuras y los esclavos comparten la otra mitad. Las tramas pueden tener 1, 3 o 5 ranuras de longitud.

La sincronización de saltos de frecuencia permite un tiempo de asentamiento de 250-260 μseg por salto para que los circuitos de radio se estabilicen. Es posible un asentamiento más rápido, pero a un mayor costo. Para una trama de una sola ranura, después del asentamiento, se desechan 366 de los 625 bits. De éstos, 126 se utilizan para un código de acceso y el encabezado, y 240 para los datos. Cuando se enlazan cinco ranuras, sólo se necesita un periodo de asentamiento y se utiliza uno ligeramente más corto, de tal manera que de los $5 \times 625 = 3125$ bits de las cinco ranuras de tiempo, 2781 se encuentran disponibles para la capa de banda base. Así, las tramas más grandes son mucho más eficientes que las de una sola ranura.

Cada trama se transmite por un canal lógico, llamado **enlace,** entre el maestro y un esclavo. Hay dos tipos de enlaces. El primero es el **ACL** (**Asíncrono no Orientado a la Conexión**), que se utiliza para datos conmutados en paquetes disponibles a intervalos irregulares. Estos datos provienen de la capa L2CAP en el nodo emisor y se entregan en la capa L2CAP en el nodo receptor. El tráfico ACL se entrega sobre la base de mejor esfuerzo. No hay garantías. Las tramas se pueden perder y tienen que retransmitirse. Un esclavo puede tener sólo un enlace ACL con su maestro.

El otro tipo de enlace es el **SCO** (**Síncrono Orientado a la Conexión**), para datos en tiempo real, como ocurre en las conexiones telefónicas. A este tipo de canal se le asigna una ranura fija en cada dirección. Por la importancia del tiempo en los enlaces SCO, las tramas que se envían a través de ellos nunca se retransmiten. En vez de ello, se puede utilizar la corrección de errores hacia delante (o corrección de errores sin canal de retorno) para conferir una confiabilidad alta. Un esclavo puede establecer hasta tres enlaces SCO con su maestro. Cada enlace de este tipo puede transmitir un canal de audio PCM de 64,000 bps.

4.6.6 La capa L2CAP de Bluetooth

La capa L2CAP tiene tres funciones principales. Primera, acepta paquetes de hasta 64 KB provenientes de las capas superiores y los divide en tramas para transmitirlos. Las tramas se reensamblan nuevamente en paquetes en el otro extremo.

Segunda, maneja la multiplexión y desmultiplexión de múltiples fuentes de paquetes. Cuando se reensambla un paquete, la capa L2CAP determina cuál protocolo de las capas superiores lo manejará, por ejemplo, el RFcomm o el de telefonía.

Tercera, la capa L2CAP se encarga de la calidad de los requerimientos de servicio, tanto al establecer los enlaces como durante la operación normal. Asimismo, durante el establecimiento de los enlaces se negocia el tamaño máximo de carga útil permitido, para evitar que un dispositivo que envíe paquetes grandes sature a uno que reciba paquetes pequeños. Esta característica es importante porque no todos los dispositivos pueden manejar paquetes de 64 KB.

4.6.7 Estructura de la trama de Bluetooth

Existen diversos formatos de trama, el más importante de los cuales se muestra en la figura 4-38. Empieza con un código de acceso que identifica al maestro, cuyo propósito es que los esclavos que se encuentren en el rango de alcance de dos maestros sepan cuál tráfico es para ellos. A continuación se encuentra un encabezado de 54 bits que contiene campos comunes de la subcapa MAC. Luego está el campo de datos, de hasta 2744 bits (para una transmisión de cinco ranuras). Para una sola ranura de tiempo, el formato es el mismo excepto que el campo de datos es de 240 bits.

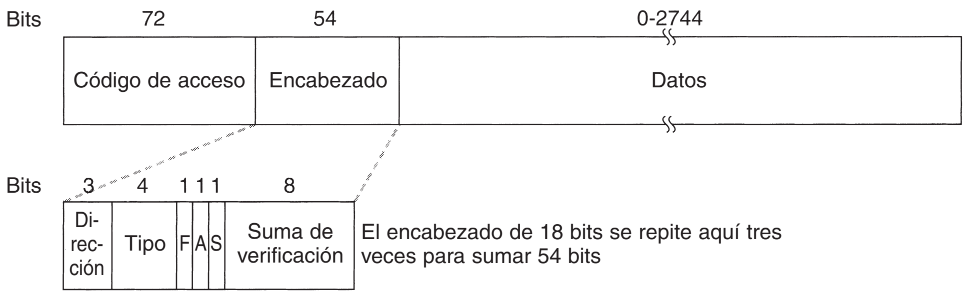


Figura 4-38. Trama de datos típica de Bluetooth.

Demos un rápido vistazo al encabezado. El campo *Dirección* identifica a cuál de los ocho dispositivos activos está destinada la trama. El campo *Tipo* indica el tipo de trama (ACL, SCO, de sondeo o nula), el tipo de corrección de errores que se utiliza en el campo de datos y cuántas ranuras de longitud tiene la trama. Un esclavo establece el bit *F* (de flujo) cuando su búfer está lleno y no puede recibir más datos. Ésta es una forma primitiva de control de flujo. El bit *A* (de confirmación de recepción) se utiliza para incorporar un ACK en una trama. El bit *S* (de secuencia) sirve para numerar las tramas con el propósito de detectar retransmisiones. El protocolo es de parada y espera, por lo que 1 bit es suficiente. A continuación se encuentra el encabezado *Suma de verificación* de 8 bits. Todo el encabezado de 18 bits se repite tres veces para formar el encabezado de 54 bits que se aprecia en la figura 4-38. En el receptor, un circuito sencillo examina las tres copias de cada bit. Si son las mismas, el bit es aceptado. De lo contrario, se impone la opinión de la mayoría. De esta forma, 54 bits de capacidad de transmisión se utilizan para enviar 10 bits de encabezado. Esto se debe a que es necesaria una gran cantidad de redundancia para enviar datos de manera confiable en un entorno con ruido mediante dispositivos de bajo costo y baja potencia (2.5 mW) con poca capacidad de cómputo.

En el campo de datos de las tramas ACL se utilizan varios formatos. Sin embargo, las tramas SCO son más sencillas: el campo de datos siempre es de 240 bits. Se definen tres variantes, que permiten 80, 160 y 240 bits de carga útil real, y el resto se utiliza para corrección de errores. En la versión más confiable (carga útil de 80 bits), el contenido se repite tres veces, al igual que el encabezado.

Debido a que el esclavo podría utilizar solamente las ranuras impares, obtiene 800 ranuras por segundo, de la misma manera que el maestro. Con una carga útil de 80 bits, la capacidad de canal del esclavo es de 64,000 bps y la del maestro también es de 64,000 bps, exactamente la necesaria para un canal de voz PCM dúplex total (razón por la cual se eligió una tasa de saltos de 1600 saltos por segundo). Estas cifras indican que un canal de voz dúplex total con 64,000 bps en cada dirección y el formato más confiable satura totalmente la *piconet* a pesar de un ancho de banda puro de 1 Mbps. En la variante menos confiable (240 bits por ranura sin redundancia a este nivel), se pueden soportar al mismo tiempo tres canales de voz dúplex total, debido a lo cual se permite un máximo de tres enlaces SCO por esclavo.

Hay mucho más por decir acerca de Bluetooth, pero ya no tenemos espacio aquí. Si desea más información, vea (Bhagwat, 2001; Bisdikian, 2001; Bray y Sturman, 2002; Haartsen, 2000; Johansson y cols., 2001; Miller y Bisdikian, 2001, y Sairam y cols., 2002).

4.7 CONMUTACIÓN EN LA CAPA DE ENLACE DE DATOS

Muchas organizaciones tienen varias LANs y desean interconectarlas. Este tipo de redes se puede conectar mediante dispositivos llamados **puentes,** que funcionan en la capa de enlace de datos. Los puentes examinan las direcciones de la capa de enlace de datos para enrutar los datos. Como no tienen que examinar el campo de carga útil de las tramas que enrutan, pueden transportar paquetes IPv4 (que se utilizan actualmente en Internet), IPv6 (que se utilizarán en el futuro en Internet), AppleTalk, ATM, OSI o de otros tipos. En contraste, los *enrutadores* examinan

las direcciones de los paquetes y realizan su trabajo de enrutamiento con base en ellas. Aunque ésta parece una clara división entre los puentes y los enrutadores, algunos desarrollos modernos, como el surgimiento de la Ethernet conmutada, han enturbiado las aguas, como veremos más tarde. En las siguientes secciones analizaremos los puentes y los conmutadores, en especial para conectar diferentes LANs 802. Para un análisis más completo sobre puentes, conmutadores y temas relacionados, vea (Perlman, 2000).

Antes de entrar de lleno a la tecnología de los puentes, vale la pena dar un vistazo a algunas situaciones comunes en las cuales se utilizan los puentes. Mencionaremos seis razones por las cuales una sola organización podría contar con varias LANs.

Primera, muchas universidades y departamentos corporativos tienen sus propias LANs, principalmente para interconectar sus propias computadoras personales, estaciones de trabajo y servidores. Dado que las metas de los distintos departamentos difieren, los departamentos escogen LANs diferentes, sin importarles lo que hagan los demás departamentos. Tarde o temprano surge la necesidad de interacción, y aquí es donde entran los puentes. En este ejemplo surgieron múltiples LANs debido a la autonomía de sus dueños.

Segunda, la organización puede estar distribuida geográficamente en varios edificios, separados por distancias considerables. Puede ser más económico tener LANs independientes en cada edificio y conectarlas mediante puentes y enlaces láser que tender un solo cable por toda la zona.

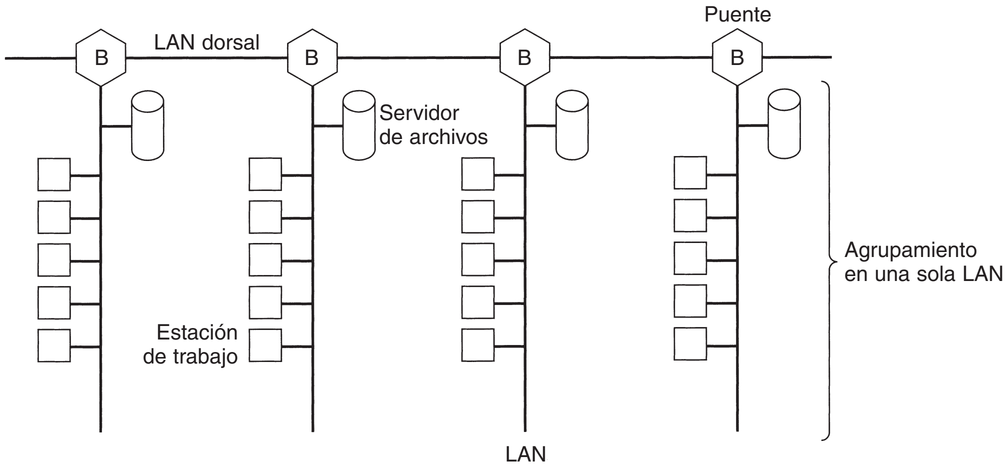


Figura 4-39. Varias LANs conectadas mediante una red dorsal para manejar una carga total mayor que la capacidad de una sola LAN.

Tercera, puede ser necesario dividir lo que lógicamente es una sola LAN en LANs individuales para manejar la carga. Por ejemplo, en muchas universidades miles de estaciones de trabajo están disponibles para los estudiantes y el cuerpo docente. Los archivos normalmente se guardan en máquinas servidoras de archivos, y son descargados a las máquinas de los usuarios a solicitud. La enorme escala de este sistema hace imposible poner todas las estaciones de trabajo en una sola

LAN, pues el ancho de banda total necesario es demasiado grande. En cambio, se usan varias LANs conectadas mediante puentes, como se muestra en la figura 4-39. Cada LAN contiene un grupo de estaciones de trabajo con su propio servidor de archivos, por lo que la mayor parte del tráfico está restringida a una sola LAN y no agrega carga a la red dorsal.

Vale la pena mencionar que aunque por lo general esquematizamos las LANs como cables con múltiples derivaciones como en la figura 4-39 (la apariencia clásica), hoy en día se implementan con mayor frecuencia mediante concentradores o conmutadores. Sin embargo, un cable largo con múltiples derivaciones para las máquinas, y un concentrador con las máquinas conectadas a él funcionan de manera idéntica. En ambos casos, todas las máquinas pertenecen al mismo dominio de colisión y todas utilizan el protocolo CSMA/CD para enviar tramas. No obstante, las LANs conmutadas son diferentes, como ya vimos y volveremos a ver en breve.

Cuarta, en algunas situaciones una sola LAN sería adecuada en términos de la carga, pero la distancia física entre las máquinas más distantes es demasiado grande (por ejemplo, mayor que 2.5 km para Ethernet). Aun si fuera fácil tender el cable, la red no funcionaría debido al retardo excesivamente grande del viaje de ida y vuelta. La única solución es segmentar la LAN e instalar puentes entre los segmentos. Con puentes, puede aumentarse la distancia física total cubierta.

Quinta, está la cuestión de la confiabilidad. En una sola LAN, un nodo defectuoso que envíe constantemente una cadena de basura echará a perder la LAN. Pueden introducirse puentes en lugares críticos, como puertas para bloquear el fuego en un edificio, y así evitar que un solo nodo enloquecido tire el sistema completo. A diferencia de un repetidor, que sólo copia lo que ve, un puente puede programarse para discriminar lo que envía y lo que no.

Sexta y última, los puentes pueden contribuir a la seguridad de la organización. La mayor parte de las interfaces LAN tienen un **modo promiscuo**, en el que *todas* las tramas se entregan a la computadora, no sólo las dirigidas a ella. Los espías y los intrusos aman esta característica. Al introducir puentes en varios lugares y tener cuidado de no reenviar tráfico delicado, es posible aislar partes de la red de manera que su tráfico no pueda escapar y caer en las manos equivocadas.

En el plano ideal, los puentes deberían ser totalmente transparentes, es decir, debería ser posible cambiar una máquina de un segmento a otro sin necesidad de modificar el hardware, software o tablas de configuración. Asimismo, debería ser posible que las máquinas de un segmento se comunicaran con las de cualquier otro segmento sin que importara el tipo de LAN que se utilizara en ambos segmentos o en los segmentos que hubiera entre ellos. Este objetivo se consigue en ocasiones, pero no siempre.

4.7.1 Puentes de 802.x a 802.y

Después de comprender por qué son necesarios los puentes, pasemos a la cuestión de su funcionamiento. En la figura 4-40 se ilustra la operación de un puente sencillo de dos puertos. El *host A* en una LAN (802.11) inalámbrica tiene un paquete por enviar a un *host* fijo, *B,* en una Ethernet (802.3) a la cual se encuentra conectada la LAN inalámbrica. El paquete desciende a la subcapa LLC y adquiere un encabezado LLC (aparece en negro en la figura). A continuación el paquete pasa a la subcapa MAC y se le antepone un encabezado 802.11 (también un terminador, que no se

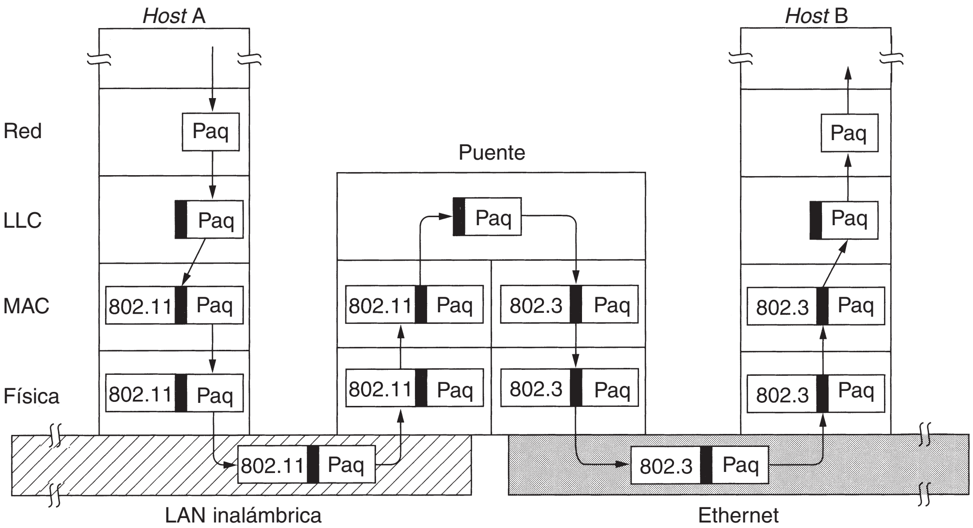


Figura 4-40. Operación de un puente entre una red 802.11 y una 802.3.

muestra en la figura). Esta unidad viaja a través del aire y es captada por la estación base, que se percata de que tiene que pasar por la Ethernet fija. Cuando el paquete llega al puente que conecta la red 802.11 con la 802.3, empieza en la capa física y realiza el recorrido hacia arriba. El encabezado 802.11 se elimina en la subcapa MAC del puente. El paquete recortado (con el encabezado LLC) se pasa a la subcapa LLC del puente. En este ejemplo, el paquete está destinado a una LAN 802.3, por lo que recorre la ruta hacia abajo en el lado 802.3 del puente y al terminar pasa a la red Ethernet. Observe que un puente que conecta *k* LANs diferentes tendrá *k* subcapas MAC diferentes y *k* capas físicas diferentes, una para cada tipo.

Hasta aquí pareciera que es muy sencillo desplazar una trama de una LAN a otra. No es así. En esta sección señalaremos algunos de los problemas que se enfrentan al intentar construir un puente entre las diversas LANs (y MANs) 802. Nos concentraremos en 802.3, 802.11 y 802.16, pero existen otras, cada una con sus propios problemas.

Para empezar, cada LAN utiliza un formato de trama distinto (vea la figura 4-41). En contraste con las diferencias entre Ethernet, token bus y token ring, que se originaron por la historia y el ego de las grandes corporaciones, aquí las diferencias son válidas hasta cierto punto. Por ejemplo, el campo *Duración* en 802.11 se justifica por el protocolo MACAW y no tiene sentido en Ethernet. En consecuencia, cualquier copia que se realice entre LANs distintas requiere de volver a dar formato, lo que lleva tiempo de CPU, una nueva suma de verificación y se presenta la posibilidad de errores sin detectar debido a bits erróneos en la memoria del puente.

Un segundo problema es que las LANs interconectadas no necesariamente operan a la misma tasa de datos. Al retransmitir una gran cantidad de tramas una tras otra, provenientes de una LAN rápida a otra más lenta, el puente será incapaz de despachar las tramas con la misma rapidez que

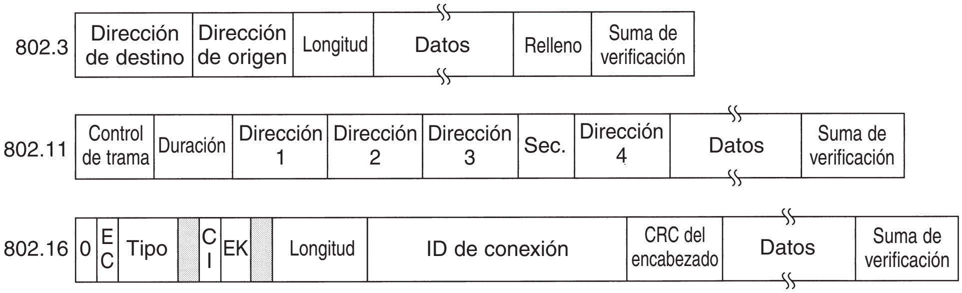


Figura 4-41. Formatos de trama de la redes 802. El dibujo no es a escala.

arriban. Por ejemplo, si una Gigabit Ethernet envía bits a su velocidad máxima a una LAN 802.11b de 11 Mbps, el puente tendrá que almacenarlos en búfer, con la esperanza de no agotar su memoria. Los puentes que conectan tres o más LANs tienen un problema similar cuando varias LANs intentan enviar datos a una misma LAN al mismo tiempo aun cuando todas operen a la misma velocidad.

Un tercer problema, y potencialmente el más grave de todos, es que distintas LANs 802 tienen diferentes longitudes máximas de trama. Un problema obvio surge cuando una trama grande tiene que reenviarse a una LAN que no puede aceptarla. En esta capa no es posible dividir la trama. Todos los protocolos dan por sentado que las tramas llegan o se pierden. No se considera el reensamblado de las tramas a partir de unidades más pequeñas. Lo anterior no significa que no se pueden diseñar tales protocolos. Es posible y se ha hecho. Es sólo que ningún protocolo de enlace de datos confiere esta característica, así que los puentes deben olvidarse de manipular la carga útil de la trama. En esencia, no hay solución. Las tramas demasiado grandes para reenviarse deben descartarse. Es suficiente sobre la transparencia.

Otro punto es la seguridad. Tanto el 802.11 como el 802.16 soportan encriptación en la capa de enlace de datos. Ethernet no. Esto significa que los diversos servicios de encriptación disponibles en las redes inalámbricas se pierden cuando el tráfico pasa sobre una Ethernet. Peor aún, si una estación inalámbrica emplea encriptación en la capa de enlace de datos, no habrá forma de desencriptar los datos cuando lleguen a la red Ethernet. Si la estación inalámbrica no utiliza encriptación, su tráfico quedará expuesto en el enlace aéreo. De cualquier manera hay un problema.

Una solución al problema de la seguridad es realizar la encriptación en una capa superior, pero en este caso la estación 802.11 tiene que saber si se está comunicando con otra estación sobre una red 802.11 (lo que significa que utilizará encriptación en la capa de enlace de datos) o con una distinta (en cuyo caso no utilizará encriptación). Al obligar a la estación a elegir se destruye la transparencia.

Un punto final es la calidad del servicio. Tanto el 802.11 como el 802.16 la ofrecen en diversas formas, el primero con el modo PCF y el último mediante conexiones a tasas de bits constantes. En Ethernet no existe el concepto de calidad del servicio, así que el tráfico proveniente de alguna de las anteriores perderá su calidad de servicio al pasar por una Ethernet.

4.7.2 Interconectividad local

La sección anterior examinó los problemas que surgen al conectar dos LANs IEEE 802 distintas mediante un solo puente. Sin embargo, en grandes organizaciones con muchas LANs, la sola interconexión entre todas da lugar a muchos problemas, aun cuando todas sean Ethernet. En un plano ideal, debería bastar con adquirir puentes diseñados para el estándar IEEE e insertar los conectores en el puente para que todo funcionara perfectamente al instante. No deberían ser necesarios cambios de hardware ni de software, ni configurar conmutadores de direcciones, descargar tablas de enrutamiento ni parámetros, nada. Tan sólo conectar los cables y empezar a trabajar. Más aún, los puentes no deberían afectar de ninguna manera el funcionamiento de LANs existentes. En otras palabras, los puentes deberían ser completamente transparentes (invisibles para todo el hardware y el software). Sorprendentemente, esto es posible. Echemos un vistazo a la manera en que se hace realidad esta magia.

En su forma más sencilla, un puente transparente funciona en modo promiscuo y acepta todas las tramas transmitidas sobre las LANs a las cuales está conectado. Tomemos como ejemplo la configuración de la figura 4-42. El puente B1 está conectado a las LANs 1 y 2, y el puente B2 está conectado a las LANs 2, 3 y 4. Una trama que llega al puente B1 en la LAN 1 con destino a *A* se puede descartar de inmediato porque se encuentra en la LAN correcta, pero una trama que llega a la LAN 1 con destino a *C* o *F* debe reenviarse.

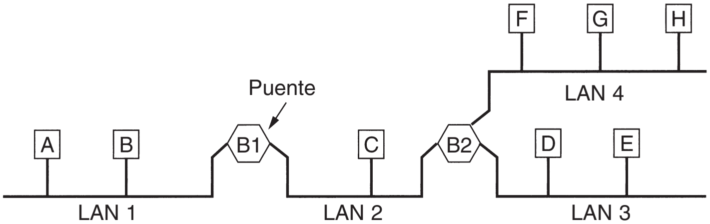


Figura 4-42. Configuración con cuatro LANs y dos puentes.

Cuando llega una trama, un puente debe decidir si la descarta o la reenvía, y si elige lo último, a cuál LAN la mandará. Esta decisión la toma consultando la dirección de destino en una enorme tabla (de *hash*) que se encuentra en su interior. La tabla lista cada posible destino e indica a cuál línea de salida (LAN) pertenece la trama. Por ejemplo, la tabla de B2 podría listar que *A* pertenece a LAN 2, ya que todo lo que B2 tiene que saber es a cuál LAN enviar las tramas para *A*. No le preocupa en absoluto el hecho de que posteriormente ocurran más reenvíos.

Cuando los puentes se conectan por primera vez, todas las tablas de *hash* están vacías. Ninguno de los puentes sabe dónde se encuentran los destinos, por lo que utilizan un algoritmo de inundación: todas las tramas que llegan con un destino desconocido se envían a todas las LANs a las cuales está conectado el puente, excepto a aquélla de la cual proceden. Con el paso del tiempo, los puentes aprenden dónde están los destinos, como se describe más adelante. Una vez conocido un destino, las tramas para él se reenvían solamente a la LAN apropiada en lugar de a todas las LANs.

El algoritmo que los puentes transparentes utilizan es **aprendizaje hacia atrás.** Como ya mencionamos, los puentes funcionan en modo promiscuo y de esta manera pueden ver todas las tramas que se envían a cualquiera de sus LANs. Al examinar la dirección del origen, pueden saber cuál máquina está disponible en cuál LAN. Por ejemplo, si el puente B1 de la figura 4-42 ve una trama proveniente de *C* en la LAN 2, sabe que es posible acceder a *C* por medio de la LAN 2, así que registra una entrada en su tabla de *hash* con la observación de que las tramas para *C* deben utilizar la LAN 2. Cualquier trama subsecuente dirigida a *C* que llegue desde la LAN 1 será reenviada, pero una trama para *C* que llegue desde la LAN 2 será descartada.

La topología puede cambiar conforme las máquinas y los puentes se enciendan y apaguen, o cuando se trasladen de un sitio a otro. Para manejar topologías dinámicas, siempre que se realiza una entrada en una tabla de *hash* se registra en la entrada la hora de llegada de una trama. Siempre que llega una trama cuyo origen ya está en la tabla, su entrada se actualiza con la hora actual. Por lo tanto, la hora asociada a cada entrada indica la última vez que se registró una trama proveniente de ese origen.

Un proceso del puente analiza periódicamente la tabla de *hash* y purga todas las entradas que tengan más de algunos minutos. De esta manera, si una computadora se desconecta de su LAN, se traslada a otro lugar del edificio y se vuelve a conectar en algún otro lugar, en pocos minutos volverá a funcionar con normalidad, sin necesidad de intervención manual. Este algoritmo también significa que si una máquina está inactiva durante algunos minutos, el tráfico destinado a ella será inundado hasta que la máquina misma envíe una trama.

El procedimiento de enrutamiento para una trama entrante depende de la LAN de que proceda (la LAN de origen) y de la LAN a la cual está destinada (la LAN de destino), como se puede ver a continuación:

1. Si las LANs de destino y de origen son la misma, descartar la trama.
2. Si las LANs de destino y de origen son diferentes, reenviar la trama.
3. Si se desconoce la LAN de destino, recurrir a la inundación.

Este algoritmo debe aplicarse cada vez que llega una trama. Chips VLSI especiales realizan la consulta y actualización de las entradas de la tabla en tan sólo algunos microsegundos.

4.7.3 Puentes con árbol de expansión

Para incrementar la confiabilidad, algunos sitios utilizan dos o más puentes en paralelo entre pares de LANs, como se muestra en la figura 4-43. Sin embargo, este arreglo también genera algunos problemas adicionales porque produce ciclos en la topología.

Un ejemplo simple de estos problemas lo tenemos al observar cómo se maneja una trama, *F*, con destino desconocido, en la figura 4-43. Cada puente, siguiendo las reglas normales para el manejo de destinos desconocidos, recurre a la inundación, que en este ejemplo es tan sólo copiar la trama a la LAN 2. Poco después, el puente 1 detecta a F_2, una trama con destino desconocido, y

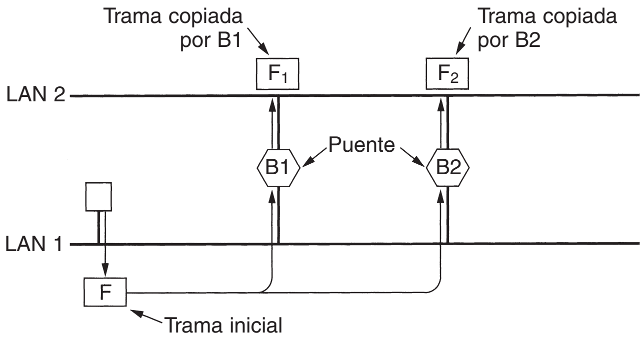


Figura 4-43. Dos puentes paralelos transparentes.

la copia a la LAN 1, lo cual genera a F_3 (no se muestra). De manera similar, el puente 2 copia a F_1 a la LAN 1 y genera a F_4 (tampoco se muestra). El puente 1 reenvía ahora a F_4 y el puente 2 copia a F_3. Este ciclo se repite una y otra vez.

La solución a este problema es que los puentes se comuniquen entre sí y cubran la topología existente con un árbol de expansión que llegue a todas las LANs. En realidad, algunas conexiones potenciales entre LANs se ignoran en el afán de construir una topología ficticia libre de ciclos. Por ejemplo, en la figura 4-44(a) vemos nueve LANs interconectadas por diez puentes. Esta configuración se puede abstraer en un grafo con las LANs como nodos. Un arco conecta dos LANs que estén unidas por un puente. El grafo puede reducirse a un árbol de expansión eliminando los arcos que se muestran como líneas punteadas en la figura 4-44(b). Con este árbol de expansión existe exactamente una ruta desde cada LAN hasta las demás LANs. Una vez que los puentes se ponen de acuerdo en el árbol de expansión, todos los reenvíos entre LANs se hacen a través del árbol de expansión. Puesto que existe sólo una ruta de cada origen a cada destino, es imposible que se generen ciclos.

Para construir el árbol de expansión, los puentes primero tienen que escoger un puente que funja como raíz del árbol. Toman esta decisión haciendo que cada uno difunda su número de serie, instalado por el fabricante y con garantía de ser único en el mundo. El puente con el menor número de serie se vuelve la raíz. A continuación, se construye un árbol con las rutas más cortas de la raíz a cada puente y LAN. Éste es el árbol de expansión. Si falla un puente o una LAN, se calcula un árbol nuevo.

El resultado de este algoritmo es que se establece una ruta única de cada LAN hasta la raíz y, por tanto, a todas las demás LANs. Aunque el árbol abarca todas las LANs, no necesariamente están presentes todos los puentes en el árbol (para evitar ciclos). Aun después de que se ha establecido el árbol de expansión, el algoritmo continúa operando a fin de detectar automáticamente cambios de topología y actualizar el árbol. El algoritmo distribuido que se usa para construir el árbol de expansión fue inventado por Radia Perlman y se describe en detalle en (Perlman, 2000). Se estandarizó en el IEEE 802.1D.

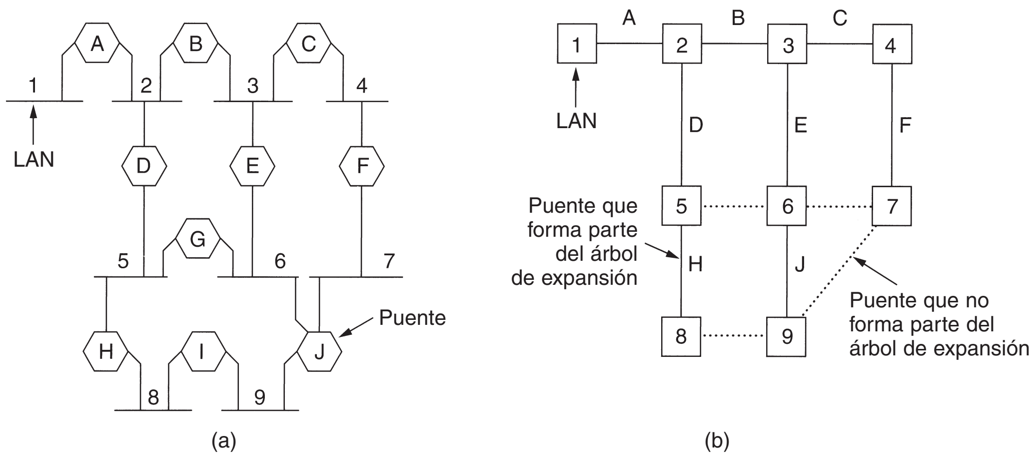


Figura 4-44. (a) LANs interconectadas. (b) Árbol de expansión que abarca las LANs. Las líneas punteadas no son parte del árbol de expansión.

4.7.4 Puentes remotos

Un uso común de los puentes es conectar dos (o más) LANs distantes. Por ejemplo, una empresa podría contar con plantas en varias ciudades, cada una con su propia LAN. En un plano ideal, todas las LANs deberían estar interconectadas de tal forma que funcionaran como una sola LAN grande.

Este objetivo se puede cumplir colocando un puente en cada LAN y conectando los puentes por pares con líneas punto a punto (por ejemplo, líneas alquiladas a una compañía telefónica). En la figura 4-45 se ilustra un sistema sencillo, con tres LANs. Aquí se aplican los algoritmos comunes de enrutamiento. La forma más sencilla de entender esto es considerar las tres líneas punto a punto como LANs sin *hosts*. A continuación tenemos un sistema normal de seis LANs interconectadas mediante cuatro puentes. En ninguna parte hasta aquí hemos dicho que una LAN debe contener *hosts*.

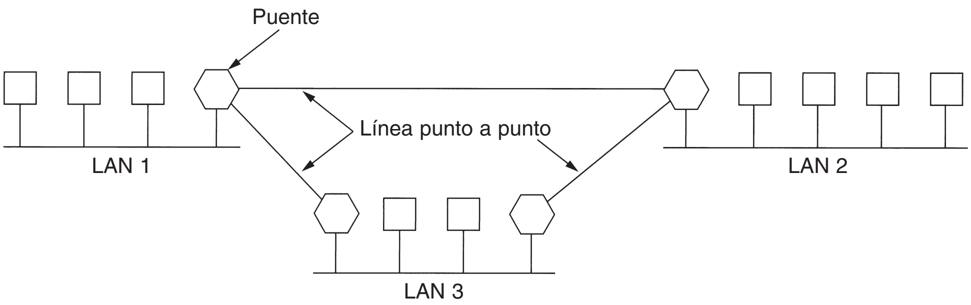


Figura 4-45. Los puentes remotos se pueden utilizar para interconectar LANs distantes.

En las líneas punto a punto se pueden utilizar diversos protocolos. Una opción es elegir algún protocolo de enlace de datos estándar de punto a punto como PPP y colocar tramas MAC completas en el campo de carga útil. Esta estrategia funciona mejor si todas las LANs son idénticas, y el único problema es conseguir que las tramas lleguen a la LAN correcta. Otra opción es eliminar tanto el encabezado como el terminador MAC en el puente de origen y agregar lo que queda en el campo de carga útil del protocolo de punto a punto. A continuación, en el puente de destino se pueden generar un nuevo encabezado y un nuevo terminador MAC. Una desventaja de este método consiste en que la suma de verificación que llega al *host* de destino no es la que se calculó en el *host* de origen, debido a lo cual tal vez no se detecten los errores ocasionados por bits defectuosos en la memoria de un puente.

4.7.5 Repetidores, concentradores, puentes, conmutadores, enrutadores y puertas de enlace

Hasta este punto hemos visto una gran variedad de formas para desplazar tramas y paquetes de un segmento de cable a otro. Hemos mencionado repetidores, puentes, conmutadores, concentradores, enrutadores y puertas de enlace. Todos estos dispositivos son de uso común, aunque difieren en formas sutiles y no tan sutiles. Puesto que son tantos, tal vez valga la pena analizarlos en conjunto para conocer sus similitudes y diferencias.

Para empezar, estos dispositivos operan en diferentes capas, como se muestra en la figura 4-46(a). La capa es importante porque los distintos dispositivos utilizan diferentes partes de información para decidir su modo de operación. En un escenario común, el usuario genera algunos datos que se enviarán a una máquina remota. Estos datos se pasan a la capa de transporte, que le agrega un encabezado, por ejemplo, un encabezado TCP, y pasa la unidad que resulta a la capa de red. Ésta incorpora su propio encabezado para obtener un paquete de capa de red, por ejemplo, un paquete IP. En la figura 4-46(b) podemos ver el paquete IP con un sombreado gris. A continuación, el paquete pasa a la capa de enlace de datos, que incorpora su propio encabezado y suma de verificación (CRC) y envía la trama resultante a la capa física para que desde ahí sea transmitida, por ejemplo, sobre una LAN.

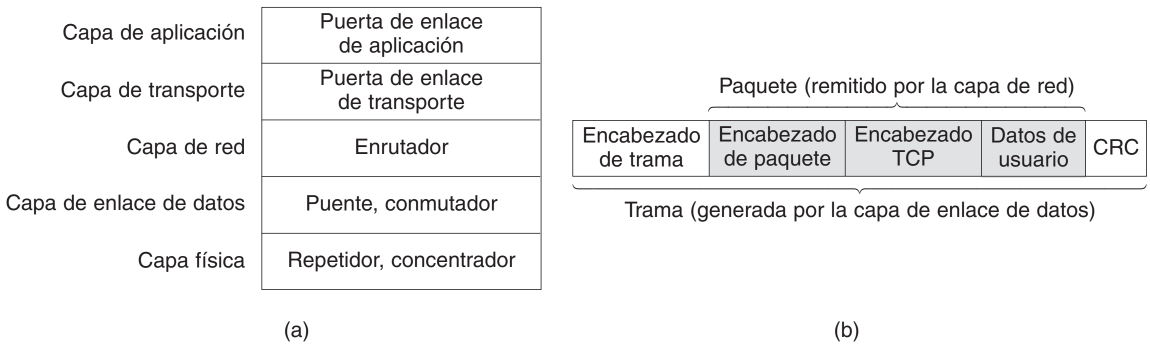


Figura 4-46. (a) Los dispositivos y sus capas correspondientes. (b) Tramas, paquetes y encabezados.

Ahora demos un vistazo a los dispositivos de conmutación y veamos cómo se relacionan con los paquetes y las tramas. Al fondo, en la capa física, se encuentran los repetidores. Éstos son dispositivos análogos conectados a dos segmentos de cable. Una señal que aparece en uno de ellos es amplificada y enviada al otro. Los repetidores no distinguen entre tramas, paquetes o encabezados. Manejan voltios. Por ejemplo, la Ethernet tradicional admite cuatro repetidores, con el propósito de extender la longitud máxima de cable de 500 a 2500 metros.

Pasemos ahora a los concentradores. Un concentrador tiene numerosos puertos de entrada que une de manera eléctrica. Las tramas que llegan a cualquiera de las líneas se envían a todas las demás. Si dos tramas llegan al mismo tiempo, chocarán, al igual que en un cable coaxial. En otras palabras, el concentrador constituye un solo dominio de colisión. Todas las líneas que convergen en un concentrador deben operar a la misma velocidad. A diferencia de los repetidores, los concentradores (por lo general) no amplifican las señales entrantes y su diseño les permite contener varias tarjetas de línea con múltiples entradas, aunque las diferencias son ligeras. Al igual que los repetidores, los concentradores no examinan las direcciones 802 ni las utilizan de ninguna manera. En la figura 4-47(a) se muestra un concentrador.

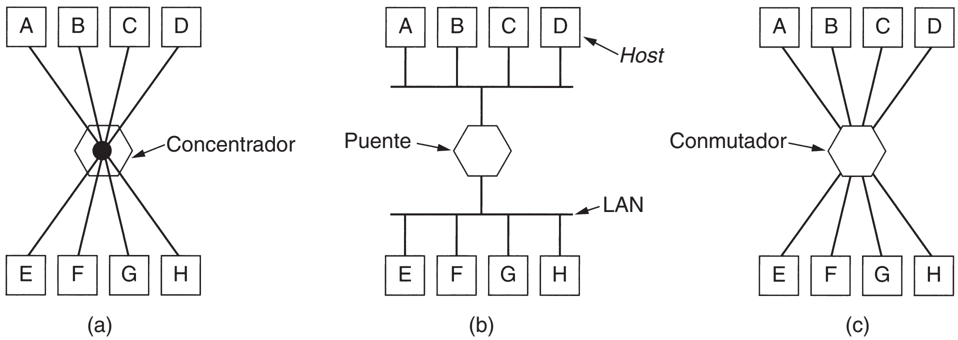


Figura 4-47. (a) Concentrador. (b) Puente. (c) Conmutador.

Veamos a continuación la capa de enlace de datos donde operan los puentes y los conmutadores. Ya hemos visto algo de los puentes. Un puente conecta dos o más LANs, como se puede ver en la figura 4-47(b). Cuando llega una trama, el software del puente extrae la dirección de destino del encabezado y la busca en una tabla para averiguar a dónde debe enviar la trama. En Ethernet, esta dirección es la dirección de destino de 48 bits que se muestra en la figura 4-17. De la misma manera que un concentrador, un puente moderno tiene tarjetas de línea, por lo general para cuatro u ocho puertos de entrada de un tipo determinado. Una tarjeta de línea para Ethernet no puede manejar tramas token ring debido a que no sabe dónde buscar la dirección de destino que viene en el encabezado de la trama. Sin embargo, un puente podría tener tarjetas de línea para diferentes tipos de red y diferentes velocidades. En contraste con un concentrador, en un puente cada puerto constituye su propio dominio de colisión.

Los conmutadores son similares a los puentes en el aspecto de que ambos enrutan tomando como base las direcciones de las tramas. De hecho, mucha gente se refiere a ellos de manera indistinta. La principal diferencia consiste en que un conmutador se utiliza con mayor frecuencia

para conectar computadoras individuales, como se puede ver en la figura 4-47(c). En consecuencia, cuando el *host A* de la figura 4-47(b) desea enviar una trama al *host B*, el puente toma la trama pero la descarta. Por el contrario, en la figura 4-47(c), el conmutador debe reenviar activamente la trama de *A* a *B* porque no existe otra forma para que ésta llegue ahí. Puesto que por lo general cada puerto del conmutador va hacia una sola computadora, éstos deben contar con espacio para muchas más tarjetas de línea que los puentes, cuyo propósito es conectar solamente LANs. Cada tarjeta de línea proporciona espacio de búfer para las tramas que llegan a sus puertos. Como cada puerto constituye su propio dominio de colisión, los conmutadores nunca pierden tramas por colisiones. Sin embargo, si las tramas llegan con mayor rapidez de la que pueden retransmitirse, el conmutador podría quedarse sin espacio de búfer y proceder a descartar tramas.

Para aliviar en parte este problema, los conmutadores modernos empiezan el reenvío de tramas tan pronto como llega el campo de encabezado del destino, antes de que el resto de la trama haya llegado (siempre y cuando el puerto de salida esté disponible, por supuesto). Estos conmutadores no utilizan la técnica de conmutación de almacenamiento y reenvío. En ocasiones se les menciona como **conmutadores cut-through.** Por lo general, este tipo de manejo se realiza por completo en hardware, en tanto que los puentes contienen tradicionalmente una CPU que realiza la conmutación de almacenamiento y reenvío en software. No obstante, debido a que todos los puentes y conmutadores modernos contienen circuitos integrados especiales para conmutación, la diferencia entre un conmutador y un puente es más un asunto de mercadotecnia que técnico.

Hasta aquí hemos visto repetidores y concentradores, que son bastante similares, así como puentes y conmutadores, que también son muy semejantes. Ahora pasaremos a los enrutadores, que son diferentes de todos los anteriores. Cuando un paquete llega a un enrutador, el encabezado y el terminador de la trama se eliminan y el paquete contenido en el campo de carga útil de la trama (sombreado en la figura 4-46) se pasa al software de enrutamiento. Este software se vale del encabezado del paquete para elegir un puerto de salida. En un paquete IP, el encabezado contendrá una dirección de 32 bits (IPv4) o 128 bits (IPv6), no una dirección 802 de 48 bits. El software de enrutamiento no analiza las direcciones de las tramas e incluso no sabe si el paquete proviene de una LAN o una línea punto a punto. En el capítulo 5 estudiaremos los enrutadores y el enrutamiento.

Una capa más arriba encontramos puertas de enlace de transporte. Estos dispositivos conectan dos computadoras que utilizan diferentes protocolos de transporte orientados a la conexión. Por ejemplo, imagine que una computadora que utiliza el protocolo TCP/IP orientado a la conexión necesita comunicarse con una computadora que emplea el protocolo de transporte ATM, también orientado a la conexión. La puerta de enlace de transporte puede copiar los paquetes de una conexión a la otra y darles el formato que necesiten.

Por último, las puertas de enlace de aplicación comprenden el formato y contenido de los datos y traducen los mensajes de un formato a otro. Por ejemplo, una puerta de enlace de correo electrónico puede traducir mensajes Internet en mensajes SMS para teléfonos móviles.

4.7.6 LANs virtuales

En los primeros días de las redes de área local, cables amarillos gruesos serpenteaban por los ductos de muchos edificios de oficinas. Conectaban a todas las computadoras por las que pasa-

ban. Con frecuencia había muchos cables, los cuales se conectaban a una red vertebral central (como en la figura 4-39) o a un concentrador central. No importaba cuál computadora pertenecía a cuál LAN. Todos los usuarios de oficinas cercanas se conectaban a la misma LAN aunque no estuvieran relacionados con ella. El aspecto geográfico se imponía al lógico.

Todo cambió con el surgimiento de 10Base-T y los concentradores en la década de 1990. El cableado de los edificios se renovó (a un costo considerable) para desechar todas las mangueras amarillas de jardín e instalar cables de par trenzado desde cada oficina hasta gabinetes centrales al final de cada pasillo o hasta salas centrales de máquinas, como se observa en la figura 4-48. Si el encargado del reemplazo del cableado era un visionario, se instalaba cable de par trenzado categoría 5; si era un simple administrador, se instalaba el cable telefónico (categoría 3) existente (que tenía que reemplazarse algunos años más tarde con la aparición de Fast Ethernet).

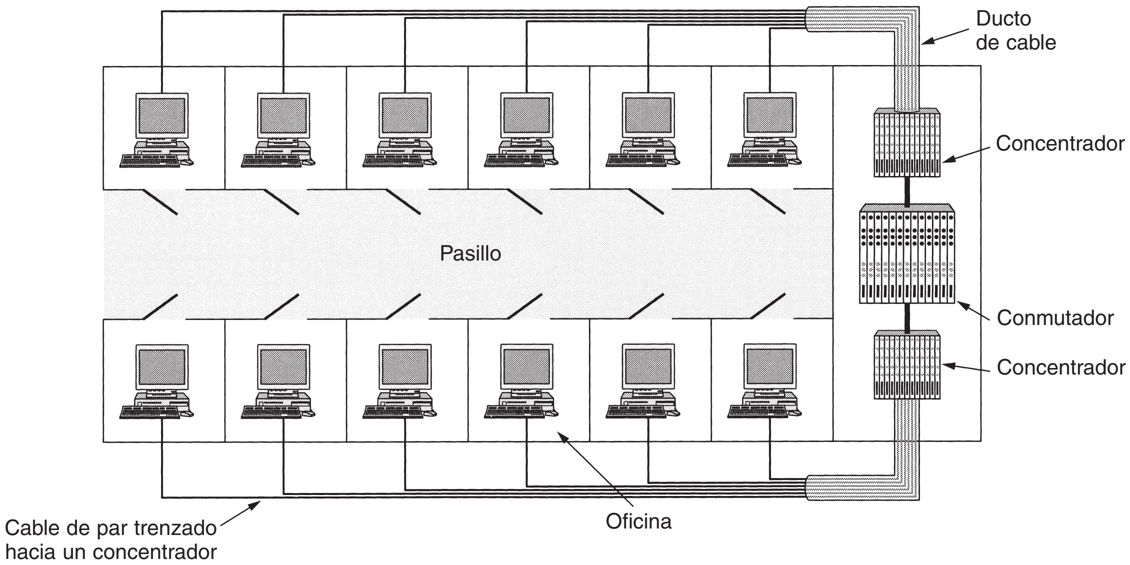


Figura 4-48. Edificio con cableado centralizado que utiliza concentradores y un conmutador.

El uso de concentradores (y de conmutadores posteriormente) con Ethernet hizo posible configurar las LANs con base en el aspecto lógico más que en el físico. Si una empresa necesita *k* LANs, compra *k* concentradores. Al elegir con cuidado qué conectores incorporar en qué concentradores, los usuarios de una LAN se pueden seleccionar de tal manera que tenga sentido para la organización, sin tomar mucho en cuenta el aspecto geográfico. Por supuesto, si dos personas del mismo departamento trabajan en diferentes edificios, es muy probable que estarán en diferentes concentradores y, en consecuencia, en diferentes LANs. No obstante, esto es mucho mejor que tener usuarios de una LAN con base totalmente en el aspecto geográfico.

¿Es importante quién está en qué LAN? Después de todo, en casi todas las organizaciones las LANs están interconectadas. Como veremos en breve, sí es importante. Por diversas razones, a los administradores de red les gusta agrupar a los usarios en LANs para reflejar la estructura de la or-

ganización más que el diseño físico del edificio. Un aspecto es la seguridad. Cualquier interfaz de red se puede operar en modo promiscuo para que copie todo el tráfico que llegue. Muchos departamentos, como los de investigación, patentes y contabilidad, manejan información que no debe salir de los límites de sus respectivas áreas. En casos como éste se justifica que todos los usuarios de un departamento sean asignados a una sola LAN y que no se permita que el tráfico salga de ésta. A los directivos no les agrada escuchar que un arreglo de este tipo sólo es posible si todos los usuarios de un departamento están en oficinas adyacentes, sin más gente entre ellos.

Un segundo aspecto es la carga. Algunas LANs se utilizan mucho más que otras, y en ocasiones podría ser necesario separarlas. Por ejemplo, si los usuarios de investigaciones realizan toda clase de experimentos que en ocasiones se les van de las manos y saturan su LAN, tal vez a los usuarios de contabilidad no les agrade tener que ceder parte de su capacidad para ayudarles.

Un tercer aspecto es la difusión. La mayoría de las LANs soporta la difusión, y muchos protocolos de la capa superior utilizan ampliamente esta característica. Por ejemplo, cuando un usuario desea enviar un paquete a una dirección IP *x*, ¿cómo sabe a cuál dirección MAC enviar la trama? En el capítulo 5 estudiaremos este asunto, pero en pocas palabras, la respuesta es que debe difundir una trama con la pregunta: ¿Quién posee la dirección IP *x*?, y esperar la respuesta. Existen muchos más ejemplos del uso de la difusión. Conforme se interconectan más y más LANs, la cantidad de difusiones (*broadcasts*) que pasan por cada máquina se incrementa de manera lineal con el número de máquinas.

Las difusiones tienen el problema asociado de que de vez en cuando las interfaces de red se averían y empiezan a generar flujos interminables de tramas de difusión. El resultado de una **tormenta de difusión** es que 1) las tramas ocupan toda la capacidad de la LAN, y 2) las máquinas de todas las LANs interconectadas se atascan procesando y descartando las tramas difundidas.

A primera vista parecería que la magnitud de las tormentas de difusión podría limitarse separando las LANs con puentes o conmutadores, pero si el objetivo es conseguir transparencia (es decir, que una máquina pueda cambiarse a una LAN distinta al otro lado del puente sin que nadie lo note), entonces los puentes tienen que reenviar las tramas difundidas.

Después de analizar por qué las empresas podrían requerir varias LANs con un alcance limitado, regresemos al problema de desacoplar la topología lógica de la física. Supongamos que un usuario es transferido de un departamento a otro de la misma empresa sin que se le cambie de oficina o que se le cambia de oficina pero no de departamento. En un entorno de concentradores con cables, cambiar al usuario a la LAN correcta implica que el administrador de la red debe desplazarse hasta el gabinete de cableado, quitar de un concentrador el conector de la máquina del usuario e insertar el mismo conector en otro concentrador.

En muchas empresas, los cambios organizacionales ocurren todo el tiempo, lo cual quiere decir que los administradores de sistemas desperdician mucho tiempo quitando y metiendo conectores de un lado a otro. Asimismo, en algunos casos el cambio no se puede realizar de ninguna manera porque el cable de par trenzado de la máquina del usuario está demasiado lejos del concentrador correcto (por ejemplo, en otro edificio).

En respuesta a la demanda de mayor flexibilidad por parte de los usuarios, los fabricantes de redes empezaron a trabajar en una forma de volver a cablear edificios completos mediante software.

El concepto que surgió se denomina **VLAN** (**LAN Virtual**) e incluso fue estandarizado por el comité 802. Ahora se encuentra funcionando en muchas organizaciones. Analicémoslo brevemente. Si desea información adicional, vea (Breyer y Riley, 1999, y Seifert, 2000).

Las VLANs se fundamentan en conmutadores especialmente diseñados para este propósito, aunque también podrían contar con algunos concentradores, como se muestra en la figura 4-48. Para configurar una red VLAN, el administrador de la red decide cuántas VLANs habrá, qué computadoras habrá en cuál VLAN y cómo se llamarán las VLANs. Es común nombrar mediante colores a las VLANs (de manera informal), ya que de esta manera es posible imprimir diagramas en color que muestren la disposición física de las máquinas, con los miembros de la LAN roja en rojo, los de la LAN verde en verde, etc. De esta forma, tanto el diseño físico como el lógico se pueden reflejar en un solo esquema.

Por ejemplo, considere las cuatro LANs de la figura 4-49(a), en la cual ocho de las máquinas pertenecen a la VLAN G (gris) y siete forman parte de la VLAN W (blanca). Dos puentes, *B1* y *B2*, conectan las cuatro LANs físicas. Si se utiliza cableado de par trenzado centralizado, también podría haber cuatro concentradores (que no se muestran), pero desde el punto de vista lógico un cable con múltiples derivaciones y un concentrador representan lo mismo. Al esquematizar la figura de esta forma se aprecia un poco menos amontonada. Asimismo, el término "puente" se emplea actualmente cuando hay varias máquinas en cada puerto, como es el caso de esta figura, pero de otra manera, "puente" y "conmutador" en esencia son indistintos. En la figura 4-49(b) se muestran las mismas máquinas y las mismas VLANs, aunque en esta ocasión con conmutadores y una sola máquina en cada puerto.

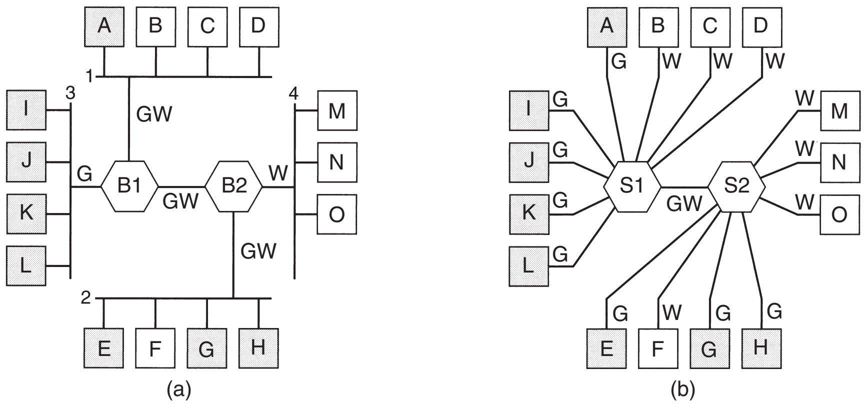


Figura 4-49. (a) Cuatro LANs físicas organizadas en dos VLANs, en gris y blanco, mediante dos puentes. (b) Las mismas 15 máquinas organizadas en dos VLANs mediante conmutadores.

Para que las VLANs funcionen correctamente, las tablas de configuración se deben establecer en los puentes o en los conmutadores. Estas tablas indican cuáles VLANs se pueden acceder a través de qué puertos (líneas). Cuando una trama llega procedente de, digamos, la VLAN gris, debe

reenviarse a todos los puertos G. De este modo se puede enviar tráfico ordinario (es decir, de unidifusión), así como de difusión y multidifusión.

Observe que un puerto puede marcarse con varios colores de VLAN. Esto se aprecia con más claridad en la figura 4-49(a). Suponga que la máquina *A* difunde una trama. El puente *B1* la recibe y detecta que proviene de una máquina de la VLAN gris, por lo cual la reenvía a todos los puertos G (excepto al puerto del que procede). Puesto que *B1* tiene sólo otros dos puertos y ambos son G, la trama se envía a ambos.

En *B2* la situación es distinta. Aquí el puerto sabe que no hay máquinas grises en la LAN 4, por lo que no envía la trama ahí. Sólo la manda a la LAN 2. Si uno de los usuarios de la LAN 4 debe cambiar de departamento y trasladarse a la VLAN gris, entonces las tablas de *B2* deben actualizarse y renombrar el puerto como GW en lugar de W. Si la máquina *F* cambia a gris, entonces el puerto de la LAN 2 debe renombrarse como G en lugar de GW.

Imaginemos ahora que todas las máquinas de las LANs 2 y 4 cambian a gris. En ese caso, no sólo los puertos de *B2* para las LANs 2 y 4 se renombran como G, sino que también el puerto de *B1* que va a *B2* debe renombrarse como G en lugar de GW porque las tramas blancas que llegan a *B1* procedentes de las LANs 1 y 3 ya no tienen que reenviarse a *B2*. En la figura 4-49(b) permanece la misma situación, sólo que aquí todos los puertos que van hacia una sola máquina se marcan con un solo color porque sólo hay una VLAN.

Hasta aquí hemos dado por sentado que los puentes y los conmutadores saben de alguna forma qué color tienen las tramas que llegan. ¿Cómo lo saben? Por medio de los tres métodos siguientes:

1. A cada puerto se le asigna un color de VLAN.
2. A cada dirección MAC se le asigna un color de VLAN.
3. A cada protocolo de la capa 3 o a cada dirección IP se le asigna un color de VLAN.

En el primer método, cada puerto se marca con un color de VLAN. Sin embargo, este método sólo funciona si todas las máquinas de un puerto pertenecen a la misma VLAN. En la figura 4-49(a), esta propiedad se aplica a *B1* para el puerto de la LAN 3 pero no al puerto de la LAN 1.

En el segundo método, el puente o el conmutador tienen una tabla con las direcciones MAC de 48 bits de cada máquina conectada a ellos, junto con la VLAN a la cual pertenece la máquina. Bajo estas condiciones, es factible mezclar VLANs en una LAN física, como en el caso de la LAN 1 de la figura 4-49(a). Cuando llega una trama, todo lo que tienen que hacer el puente o el conmutador es extraer la dirección MAC y buscarla en una tabla para averiguar de qué VLAN proviene.

En el tercer método el puente o el conmutador examinan el campo de carga útil de la trama, por ejemplo, para clasificar todas las máquinas IP en una VLAN y todas las máquinas AppleTalk en otra. En el primer caso, la dirección IP se puede utilizar también para identificar a la máquina. Esta estrategia es más útil cuando varias máquinas son computadoras portátiles que se pueden acoplar en cualquiera de diversos lugares. Puesto que cada estación de acoplamiento tiene su propia dirección MAC, el solo hecho de saber cuál estación de acoplamiento se utilizó no indica en absoluto en cuál VLAN se encuentra la computadora portátil.

El único problema de este enfoque es que transgrede la regla más elemental de la conectividad: independencia de las capas. A la capa de enlace de datos no le incumbe lo que esté en el campo de carga útil. No le corresponde analizar la carga útil ni tomar decisiones con base en el contenido. Una consecuencia del uso de este enfoque es que un cambio en el protocolo de la capa 3 (por ejemplo, una actualización de IPv4 a IPv6) ocasiona que los conmutadores fallen repentinamente. Por desgracia, en el mercado hay conmutadores que funcionan de esta manera.

Por supuesto, no hay nada de malo en enrutar con base en las direcciones IP —casi todo el capítulo 5 está dedicado al enrutamiento IP— pero al mezclar las capas se pueden propiciar problemas. Un fabricante de conmutadores podría minimizar esta situación argumentando que sus conmutadores comprenden tanto IPv4 como IPv6, así que no hay problema. ¿Pero qué pasará cuando surja IPv7? En tal caso, el fabricante tal vez dirá: Compre nuevos conmutadores, ¿cuál es el problema?

El estándar IEEE 802.1Q

Al ahondar un poco más en este asunto salta a la vista que lo importante es la VLAN de la trama, no la VLAN de la máquina emisora. Si hubiera alguna forma de identificar la VLAN en el encabezado de la trama, se desvanecería la necesidad de examinar la carga útil. Para una LAN nueva como 802.11 u 802.16 habría sido bastante fácil tan sólo agregar un campo de VLAN en el encabezado. De hecho, el campo *Identificador de conexión* del estándar 802.16 es muy parecido a un identificador VLAN. ¿Pero qué se puede hacer con Ethernet, que es la LAN dominante y no tiene campos disponibles para el identificador VLAN?

El comité IEEE 802 se enfrentó a este problema en 1995. Después de muchas discusiones, hizo lo impensable y cambió el encabezado de Ethernet. El nuevo formato se publicó en el estándar **802.1Q** del IEEE, emitido en 1998. El nuevo formato contiene una etiqueta VLAN; en breve la examinaremos. No es de sorprender que el cambio de algo ya bien establecido como el encabezado de Ethernet no sea nada sencillo. Algunas de las preguntas que surgen son:

1. ¿Tenemos que tirar a la basura cientos de millones de tarjetas Ethernet existentes?
2. Si no es así, ¿quién generará los nuevos campos?
3. ¿Qué sucederá con las tramas que ya tienen el tamaño máximo?

Por supuesto, el comité 802 estaba consciente de estos problemas y tenía que encontrar soluciones, lo cual hizo.

La clave para la solución consiste en comprender que los campos VLAN sólo son utilizados por los puentes y los conmutadores, no por las máquinas de los usuarios. De ahí que en la figura 4-49 no sea realmente necesario que estén presentes en las líneas que van hacia las estaciones finales siempre y cuando se encuentren en la línea entre los puentes o los conmutadores. Así, para utilizar VLANs, los puentes o los conmutadores deben tener soporte para VLAN, pero ese ya era un requisito. Ahora sólo estamos agregando el requisito adicional de que tengan soporte para 802.1Q, requisito que los nuevos ya cubren.

Respecto a la cuestión de si es necesario desechar todas las tarjetas Ethernet existentes, la respuesta es no. Recuerde que el comité 802.3 no pudo conseguir que la gente cambiara el campo *Tipo* por un campo *Longitud.* Ya podrá imaginar la reacción ante el anuncio de que todas las tarjetas Ethernet existentes tuvieran que desecharse. Sin embargo, se espera que las nuevas tarjetas Ethernet que salgan al mercado tendrán compatibilidad con el 802.1Q y llenarán correctamente el campo VLAN.

Por lo tanto, si el emisor no generará los campos VLAN, ¿quién lo hará? La respuesta es que el primer puente o conmutador con soporte de VLAN en recibir una trama los agregará y el último que los reciba los eliminará. ¿Pero cómo sabrán cuál trama corresponde a cuál VLAN? Bueno, el primer puente o conmutador podría asignar un número de VLAN a un puerto, analizar la dirección MAC o (¡Dios no lo quiera!) examinar la carga útil. Mientras todas las tarjetas Ethernet no se apeguen al estándar 802.1Q, estaremos en donde empezamos. La esperanza real es que todas las tarjetas Gigabit Ethernet se apegarán a 802.1Q desde el principio y que en tanto la gente se actualiza a Gigabit Ethernet, el 802.1Q se introducirá automáticamente. En cuanto al problema de las tramas mayores a 1518 bytes, el 802.1Q tan sólo incrementó el límite a 1522 bytes.

Durante el proceso de transición, muchas instalaciones tendrán algunas máquinas heredadas (en su mayoría, clásicas o Fast Ethernet) que no soportarán VLAN y otras (por lo general, Gigabit Ethernet) que sí lo harán. Esta situación se ilustra en la figura 4-50, en donde los símbolos sombreados representan máquinas que soportan VLAN y los vacíos no las soportan. Por simplicidad, damos por sentado que todos los conmutadores soportan VLAN. Si no es así, el primer conmutador que soporte VLAN puede incorporar las etiquetas con base en las direcciones MAC o IP.

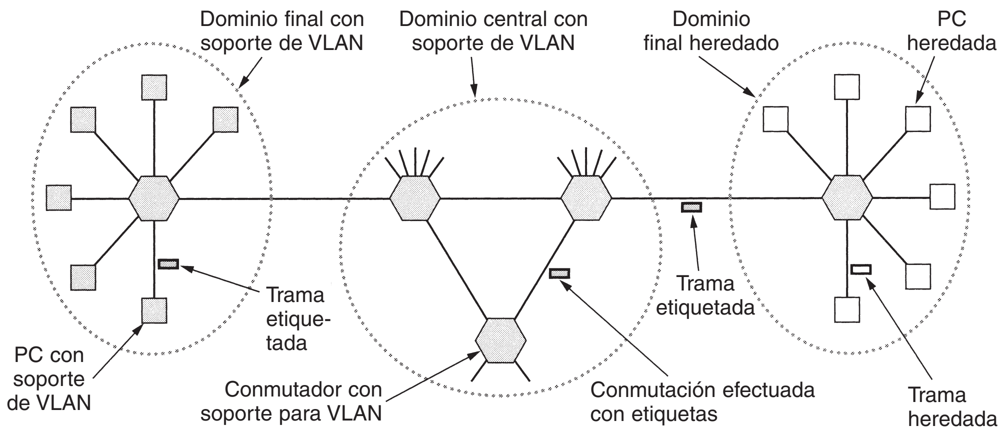


Figura 4-50. Transición de Ethernet heredada a Ethernet con soporte para VLAN. Los símbolos sombreados representan soporte para VLAN, a diferencia de los vacíos.

En esta figura, las tarjetas Ethernet con soporte para VLAN generan directamente tramas etiquetadas (es decir, 802.1Q) y la conmutación posterior se vale de estas etiquetas. Para realizar esta conmutación, los conmutadores tienen que saber cuáles VLANs están al alcance en cada puerto, lo mismo que antes. El hecho de saber que una trama pertenece a la VLAN gris no es de mucha

ayuda sino hasta que el conmutador sabe cuáles puertos tienen conexión con las máquinas de la VLAN gris. De esta forma, el conmutador necesita una tabla indexada por VLAN que le indique cuáles puertos puede utilizar y si éstos tienen soporte para VLAN o son heredados.

Cuando una PC heredada envía una trama a un conmutador con soporte para VLAN, el conmutador genera una trama etiquetada apoyándose en el conocimiento que tiene de la VLAN del emisor (utilizando el puerto, la dirección MAC o la dirección IP). De ahí en adelante, no importa que el emisor sea una máquina heredada. Asimismo, un conmutador que necesita entregar una trama etiquetada a una máquina heredada tiene que darle a la trama el formato heredado antes de entregarla.

Demos ahora un vistazo al formato de la trama 802.1Q, que se muestra en la figura 4-51. El único cambio es la adición de un par de campos de dos bytes. El primero es la *ID del protocolo de VLAN.* Siempre tiene el valor 0x8100. Como esta cifra es mayor que 1500, todas las tarjetas Ethernet lo interpretan como un tipo más que como una longitud. Lo que una tarjeta heredada hace con una trama como ésta es discutible porque dichas tramas no deberían enviarse a tarjetas heredadas.

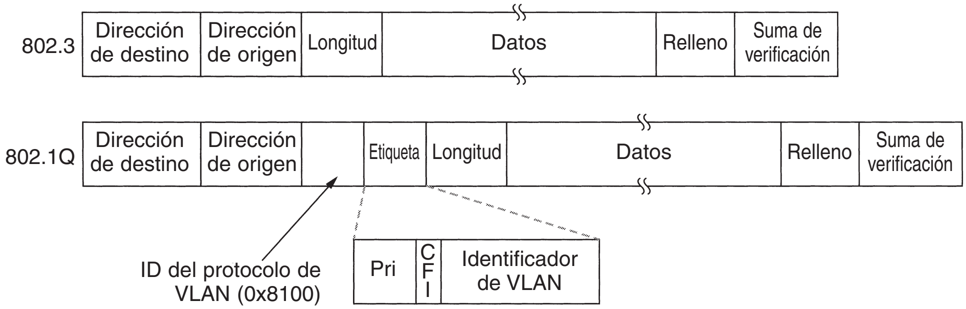


Figura 4-51. Formatos de trama Ethernet 802.3 (heredada) y 802.1Q.

El segundo campo de dos bytes contiene tres subcampos. El principal es el *Identificador de VLAN*, que ocupa los 12 bits de orden menor. Éste es el punto central de la cuestión: ¿a qué VLAN pertenece la trama? El campo *Prioridad* de 3 bits no tiene absolutamente nada que ver con las VLANs, pero como el cambio del encabezado Ethernet es un suceso poco frecuente que tarda tres años y ocupa a un ciento de personas, ¿por qué no incorporarle algunas otras cosas buenas en el proceso? Este campo permite distinguir el tráfico en tiempo real estricto del tráfico en tiempo real flexible y del tráfico no sensible al retardo, con el propósito de ofrecer una mejor calidad de servicio sobre Ethernet. Esto es necesario para el transporte de voz sobre Ethernet (aunque, para ser justos, IP tiene un campo similar desde hace un cuarto de siglo y nadie lo utiliza).

El último bit, *CFI* (*Indicador del Formato Canónico*), debió haberse llamado *CEI* (*Indicador del Ego Corporativo*). Su propósito original era indicar las direcciones MAC *little endian* en comparación con las *big endian*, pero esto ha cambiado con el tiempo. Actualmente indica que la carga útil contiene una trama 802.5 congelada-seca que se espera encuentre otra LAN 802.5 en el destino cuando se transmita a través de Ethernet. Por supuesto, este arreglo no tiene absolutamente

nada que ver con las VLANs. Pero la política de los comités de estándares no difiere mucho de la política común: si votas por mi bit, votaré por el tuyo.

Como ya mencionamos, cuando una trama etiquetada llega a un conmutador con soporte para VLAN, éste utiliza el ID de la VLAN como índice de tabla para averiguar a cuáles puertos enviar la trama. ¿Pero de dónde proviene la tabla? Si se elabora en forma manual, tenemos que empezar desde cero: la configuración manual de los puentes. La ventaja de los puentes transparentes es que son *plug and play* y no requieren configuración manual. Sería un terrible retroceso perder esa propiedad. Por fortuna, los puentes con soporte para VLAN se pueden autoconfigurar al observar las etiquetas que arriben a ellos. Si una trama etiquetada como VLAN 4 llega al puerto 3, entonces aparentemente una máquina en el puerto 3 se encuentra en la VLAN 4. El estándar 802.1Q explica cómo construir las tablas de manera dinámica, en su mayor parte referenciando porciones apropiadas del algoritmo de Perlman estandarizado en el 802.1D.

Antes de abandonar el tema del enrutamiento para VLAN, vale la pena hacer una última observación. Mucha gente de Internet y Ethernet es fanática de las redes no orientadas a la conexión y se oponen terminantemente a todo lo que huela a conexiones en las capas de enlace de datos y de red. No obstante, las VLANs incluyen un aspecto que es sorprendentemente similar a una conexión. Para utilizar las VLANs de manera apropiada, cada trama lleva un identificador especial nuevo que se utiliza como índice en una tabla dentro del conmutador para averiguar el destino al que se debe enviar la trama. Esto es precisamente lo que se hace en las redes orientadas a la conexión. En las redes no orientadas a la conexión, la dirección de destino es la que se utiliza para el enrutamiento, no un tipo de identificador de conexión. En el capítulo 5 abundaremos en este conexionismo gradual.

4.8 RESUMEN

Algunas redes tienen un solo canal que se usa para todas las comunicaciones. En estas redes, el aspecto clave del diseño es la asignación del canal entre las estaciones competidoras que desean usarlo. Se han desarrollado muchos algoritmos de asignación de canal. En la figura 4-52 se presenta un resumen de algunos de los métodos de asignación de canal más importantes.

Los métodos de asignación más sencillos son la FDM y la TDM; son eficientes con un número de estaciones pequeño y fijo y tráfico continuo. Ambos esquemas se usan ampliamente en estas circunstancias; por ejemplo, para dividir el ancho de banda de las troncales telefónicas.

Con un número grande y variable de estaciones, o con un tráfico en ráfagas, la FDM y la TDM son soluciones pobres. Se ha propuesto como alternativa el protocolo ALOHA, con y sin ranuras y control. El ALOHA y sus muchas variantes y derivaciones han sido ampliamente estudiados, analizados y usados en sistemas reales.

Cuando puede detectarse el estado del canal, las estaciones pueden evitar el comienzo de una transmisión mientras otra estación está transmitiendo. Esta técnica, la detección de portadora, ha producido varios protocolos que pueden usarse en LANs y MANs.

Método	Descripción
FDM	Dedica una banda de frecuencia a cada estación
WDM	Esquema FDM dinámico para fibra
TDM	Dedica una ranura de tiempo a cada estación
ALOHA puro	Transmisión asíncrona en cualquier momento
ALOHA ranurado	Transmisión aleatoria en ranuras de tiempo bien definidas
CSMA persistente-1	Acceso múltiple con detección de portadora estándar
CSMA no persistente	Retardo aleatorio cuando se detecta que el canal está ocupado
CSMA persistente-p	CSMA, pero con una probabilidad de persistencia p
CSMA/CD	CSMA, pero aborta al detectar una colisión
Mapa de bits	Calendarización *round robin* mediante mapa de bits
Conteo descendente binario	La estación disponible con el número más alto toma el turno
Recorrido de árbol	Contención reducida mediante habilitación selectiva
MACA, MACAW	Protocolos de LAN inalámbrica
Ethernet	CSMA/CD con retraso exponencial binario
FHSS	Espectro disperso con salto de frecuencia
DSSS	Espectro disperso de secuencia directa
CSMA/CA	Acceso múltiple con detección de portadora y evitación de colisiones

Figura 4-52. Métodos de asignación de canal y sistemas para canal común.

Se conoce una clase de protocolos que eliminan por completo la contención, o cuando menos la reducen considerablemente. El conteo binario descendente elimina por completo la contención. El protocolo de recorrido de árbol la reduce dividiendo dinámicamente las estaciones en dos grupos separados, uno que puede transmitir y otro que no. Se intenta hacer la división de tal manera que sólo una estación lista para transmitir pueda hacerlo.

Las LANs inalámbricas tienen sus propios problemas y soluciones. El problema principal lo causan las estaciones ocultas, por lo que el CSMA no funciona. Una clase de soluciones, tipificadas por MACA y MACAW, intenta estimular las transmisiones en las cercanías del destino, para hacer que el CSMA funcione mejor. También se usan el espectro disperso con salto de frecuencia y el espectro disperso de secuencia directa. El IEEE 802.11 combina CSMA y MACAW para producir CSMA/CA.

Ethernet predomina en el campo de las redes de área local. Utiliza CSMA/CD para la asignación de canal. Las primeras versiones empleaban un cable que serpenteaba entre las máquinas, pero en la actualidad son más comunes los cables de par trenzado hacia concentradores y conmutadores. Las velocidades se han incrementado de 10 Mbps a 1 Gbps y siguen en aumento.

Las LANs inalámbricas se están popularizando, y el 802.11 domina el campo. Su capa física permite cinco diferentes modos de transmisión, entre ellos el infrarrojo, diversos esquemas de

espectro disperso y un sistema FDM multicanal con una estación base en cada celda, aunque también puede funcionar sin ninguna. El protocolo es una variante de MACAW, con detección de portadora virtual.

Las MANs inalámbricas están empezando a aparecer. Son sistemas de banda amplia que utilizan radio para reemplazar la última milla en conexiones telefónicas. También utilizan técnicas tradicionales de modulación de banda estrecha. La calidad de servicio es importante, y el estándar 802.16 define cuatro clases (tasa de bits constante, dos tasas variables de bits y una de mejor esfuerzo).

El sistema Bluetooth también es inalámbrico, aunque está más enfocado a los sistemas de escritorio, para conectar diademas telefónicas y otros periféricos a las computadoras sin necesidad de cables. También se utiliza para conectar periféricos, como máquinas de fax, a los teléfonos móviles. Al igual que el 801.11, utiliza espectro disperso con saltos de frecuencia en la banda ISM. Debido al nivel de ruido esperado en muchos entornos y a la necesidad de interacción en tiempo real, sus diversos protocolos incorporan una sofisticada corrección de errores hacia delante.

Con tantas LANs diferentes, es necesaria una forma para interconectarlas. Los puentes y los conmutadores tienen este propósito. El algoritmo de árbol de expansión se utiliza para construir puentes *plug and play*. La VLAN es un nuevo desarrollo del mundo de la interconexión de LANs, que separa la topología lógica de la topología física de las LANs. Se ha introducido un nuevo formato para las tramas Ethernet (802.1Q), cuyo propósito es facilitar la utilización de las VLANs en las organizaciones.

PROBLEMAS

1. Para este problema, utilice una fórmula de este capítulo, pero primero enúnciela. Las tramas arriban de manera aleatoria a un canal de 100 Mbps para su transmisión. Si el canal está ocupado cuando arriba una trama, ésta espera su turno en una cola. La longitud de la trama se distribuye exponencialmente con una media de 10,000 bits/trama. Para cada una de las siguientes tasas de llegada de tramas, dé el retardo experimentado por la trama promedio, incluyendo tanto el tiempo de encolamiento como el de transmisión.
(a) 90 tramas/seg.
(b) 900 tramas/seg.
(c) 9000 tramas/seg.

2. Un grupo de *N* estaciones comparte un canal ALOHA puro de 56 kbps. La salida de cada estación es una trama de 1000 bits en promedio cada 100 seg aun si la anterior no ha sido enviada (por ejemplo, las estaciones pueden almacenar en búfer las tramas salientes). ¿Cuál es el valor máximo de *N*?

3. Considere el retardo del ALOHA puro comparándolo con el ALOHA ranurado cuando la carga es baja. ¿Cuál es menor? Explique su respuesta.

4. Diez mil estaciones de reservaciones de una aerolínea compiten por un solo canal ALOHA ranurado. La estación promedio hace 18 solicitudes/hora. Una ranura dura 125 μseg. ¿Cuál es la carga aproximada total del canal?

5. Una gran población de usuarios de ALOHA genera 50 solicitudes/seg incluidas tanto originales como retransmisiones. El tiempo se divide en ranuras de 40 mseg.
(a) ¿Cuál es la oportunidad de éxito en el primer intento?
(b) ¿Cuál es la probabilidad exacta de *k* colisiones y después tener éxito?
(c) ¿Cuál es el número esperado de intentos de transmisión necesarios?

6. Mediciones en un canal ALOHA ranurado con una cantidad infinita de usuarios muestra que 10% de las ranuras están inactivas.
(a) ¿Qué carga, G, tiene el canal?
(b) ¿Cuál es la velocidad real de transporte?
(c) ¿El canal está subcargado o sobrecargado?

7. En un sistema ALOHA ranurado de población infinita, la cantidad media de ranuras que espera una estación entre una colisión y su retransmisión es de 4. Grafique la curva de retardo contra velocidad real de transporte de este sistema.

8. ¿Cuánto debe esperar una estación, *s*, en el peor de los casos, antes de empezar a transmitir su trama sobre una LAN que utiliza
(a) el protocolo básico de mapa de bits?
(b) el protocolo de Mok y Ward con cambio de números virtuales de estación?

9. Una LAN usa la versión de Mok y Ward del conteo descendente binario. En cierto momento, las 10 estaciones tienen los números de estación virtual 8, 2, 4, 5, 1, 7, 3, 6, 9 y 0. Las tres estaciones siguientes que van a enviar son: 4, 3 y 9, en ese orden. ¿Cuáles son los nuevos números de estación virtual una vez que las tres han terminado sus transmisiones?

10. Dieciséis estaciones contienden por un canal compartido que usa el protocolo de recorrido de árbol. Si todas las estaciones cuyas direcciones son números primos de pronto quedaran listas al mismo tiempo, ¿cuántas ranuras de bits se necesitan para resolver la contención?

11. Un conjunto de 2^n estaciones usa el protocolo de recorrido de árbol adaptable para arbitrar el acceso a un cable compartido. En cierto momento, dos de ellas quedan listas. ¿Cuál es el número de ranuras mínimo, máximo y medio para recorrer el árbol si $2^n \gg 1$?

12. Las LANs inalámbricas que estudiamos usaban protocolos como MACA en lugar de CSMA/CD. ¿En qué condiciones sería posible usar CSMA/CD?

13. ¿Qué propiedades tienen en común los protocolos de acceso a canal WDMA y GSM? Consulte el GSM en el capítulo 2.

14. Seis estaciones, de *A* a *F*, se comunican mediante el protocolo MACA. ¿Es posible que dos transmisiones tengan lugar de manera simultánea? Explique su respuesta.

15. Un edificio de oficinas de siete pisos tiene 15 oficinas adyacentes por piso. Cada oficina contiene un enchufe de pared para una terminal en la pared frontal, por lo que los enchufes forman una retícula triangular en el plano vertical, con una separación de 4 m entre enchufes, tanto vertical como horizontalmente. Suponiendo que es factible tender un cable recto entre cualquier par de enchufes, horizontal, vertical o diagonalmente, ¿cuántos metros de cable se necesitan para conectar todos los enchufes usando
(a) una configuración en estrella con un solo enrutador en medio?
(b) una LAN 802.3?

16. ¿Cuál es la tasa de baudios de la Ethernet de 10 Mbps estándar?

17. Bosqueje la codificación Manchester para el flujo de bits: 0001110101.

18. Bosqueje la codificación diferencial Manchester para el flujo de bits del problema anterior. Suponga que la línea se encuentra inicialmente en el estado bajo.

19. Una LAN CSMA/CD (no la 802.3) de 10 Mbps y 1 km de largo tiene una velocidad de propagación de 200 m/μseg. En este sistema no se permiten los repetidores. Las tramas de datos tienen 256 bits de longitud, incluidos 32 bits de encabezado, suma de verificación y un poco más de sobrecarga. La primera ranura de bits tras una transmisión exitosa se reserva para que el receptor capture el canal y envíe una trama de confirmación de recepción de 32 bits. ¿Cuál es la tasa de datos efectiva, excluyendo la sobrecarga, suponiendo que no hay colisiones?

20. Dos estaciones CSMA/CD intentan transmitir archivos grandes (multitrama). Tras el envío de cada trama, contienden por el canal usando el algoritmo de retroceso exponencial binario. ¿Cuál es la probabilidad de que la contención termine en la ronda *k*, y cuál es la cantidad media de rondas por periodo de contención?

21. Considere la construcción de una red CSMA/CD que opere a 1 Gbps a través de un cable de 1 km de longitud sin repetidores. La velocidad de la señal en el cable es de 200,000 km/seg. ¿Cuál es el tamaño mínimo de trama?

22. Un paquete IP que se transmitirá a través de Ethernet tiene 60 bytes de longitud, incluyendo todos los encabezados. Si no se utiliza LLC, ¿se necesita relleno en la trama Ethernet, y de ser así, cuántos bytes?

23. Las tramas Ethernet deben tener al menos 64 bytes de longitud para asegurar que el transmisor permanezca en línea en caso de que ocurra una colisión en el extremo más lejano del cable. Fast Ethernet tiene el mismo tamaño mínimo de trama de 64 bytes pero puede recibir los bits diez veces más rápido. ¿Cómo es posible mantener el mismo tamaño mínimo de trama?

24. Algunos libros citan que el tamaño máximo de una trama Ethernet es de 1518 bytes en lugar de 1500. ¿Están en un error? Explique su respuesta.

25. La especificación 1000Base-SX indica que el reloj debe correr a 1250 MHz, aun cuando se supone que Gigabit Ethernet funciona a 1 Gbps. ¿Esta velocidad más alta confiere un margen adicional de seguridad? Si no es así, ¿qué está pasando?

26. ¿Cuántas tramas por segundo puede manejar Gigabit Ethernet? Reflexione con cuidado y tome en cuenta todos los casos relevantes. *Sugerencia:* es importante el hecho de que se trata de *Gigabit* Ethernet.

27. Mencione dos redes que permitan empaquetar tramas una tras otra. ¿Por qué es importante esta característica?

28. En la figura 4.27 se muestran cuatro estaciones, *A, B, C* y *D.* ¿Cuál de las dos últimas estaciones cree que está más cerca de *A* y por qué?

29. Suponga que una LAN 802.11b de 11 Mbps transmite tramas de 64 bytes una tras otra sobre un canal de radio con una tasa de error de 10^{-7}. ¿Cuántas tramas por segundo en promedio resultarán dañadas?

30. Una red 802.16 tiene un ancho de canal de 20 MHz. ¿Cuántos bits/seg se pueden enviar a una estación suscrita?

31. El IEEE 802.16 soporta cuatro clases de servicio. ¿Cuál es la mejor clase de servicio para enviar vídeo sin comprimir?

32. Dé dos razones por las cuales las redes podrían usar un código de corrección de errores en lugar de detección de errores y retransmisión.

33. En la figura 4-35 podemos ver que un dispositivo Bluetooth puede estar en dos *piconets* al mismo tiempo. ¿Hay alguna razón por la cual un dispositivo no pueda fungir como maestro en ambas al mismo tiempo?

34. La figura 4-25 muestra varios protocolos de capa física. ¿Cuál de éstos está más cercano al protocolo de capa física Bluetooth? ¿Cuál es la principal diferencia entre ambos?

35. Bluetooth soporta dos tipos de enlaces entre un maestro y un esclavo. ¿Cuáles son y para qué se utiliza cada uno?

36. Las tramas de *beacon* en el espectro disperso con salto de frecuencia, variante del 802.11, contienen el tiempo de permanencia. ¿Cree que las tramas de *beacon* análogas de Bluetooth también contienen tiempo de permanencia? Explique su respuesta.

37. Considere las LANs interconectadas que se muestran en la figura 4-44. Suponga que los *hosts a* y *b* se encuentran en la LAN 1, *c* está en la LAN 2 y *d* está en la LAN 8. En principio, las tablas de *hash* de todos los puentes están vacías y se utiliza el árbol de expansión que se muestra en la figura 4-44(b). Muestre la manera en que cambian las tablas de *hash* de los diversos puentes después de que cada uno de los siguientes sucesos ocurren en secuencia, primero (a) y a continuación (b), y así sucesivamente.
(a) *a* envía a *d*.
(b) *c* envía a *a*.
(c) *d* envía a *c*.
(d) *d* pasa a la LAN 6.
(e) *d* envía a *a*.

38. Una consecuencia del uso de un árbol de expansión para reenviar tramas en una LAN extendida es que algunos puentes tal vez no participen en absoluto en el reenvío de tramas. Identifique tres puentes que se encuentren en esta situación en la figura 4-44. ¿Hay alguna razón para conservar estos puentes, aun cuando no se utilicen para el reenvío?

39. Suponga que un conmutador tiene tarjetas de línea para cuatro líneas de entrada. Con frecuencia, una trama que llega en una de las líneas tiene que salir en otra línea de la misma tarjeta. ¿A qué decisiones se enfrenta el diseñador del conmutador como resultado de esta situación?

40. Un conmutador diseñado para Fast Ethernet tiene una tarjeta madre que puede transportar 10 Gbps. ¿Cuántas tramas/seg puede manejar en el peor de los casos?

41. Considere la red de la figura 4-49(a). Si la máquina *J* tuviera que volverse blanca repentinamente, ¿serían necesarios cambios para el etiquetado? Si es así, ¿cuáles?

42. Describa brevemente la diferencia entre los conmutadores de almacenamiento y reenvío y los *cut-through*.

43. Los conmutadores de almacenamiento y reenvío tienen una ventaja sobre los *cut-through* en relación con las tramas dañadas. Explique cuál es.

44. Las tablas de configuración son necesarias en los conmutadores y los puentes para que las VLANs funcionen. ¿Qué pasaría si las VLANs de la figura 4-49(a) utilizaran concentradores en vez de cables con múltiples derivaciones? ¿Los concentradores también necesitarían tablas de configuración? ¿Por qué sí o por qué no?

45. En la figura 4-50 el conmutador del dominio final heredado en la parte derecha tiene soporte para VLAN. ¿Sería posible utilizar ahí un conmutador heredado? Si es así, ¿cómo funcionaría? En caso contrario, ¿por qué no?

46. Escriba un programa para simular el comportamiento del protocolo CSMA/CD sobre Ethernet cuando hay *N* estaciones listas para transmitir en el momento en que se está transmitiendo una trama. El programa deberá informar las veces que cada estación inicia exitosamente el envío de su trama. Suponga que un pulso de reloj ocurre una vez cada ranura de tiempo (51.2 microsegundos) y que la detección de una colisión y el envío de una secuencia atorada tarda una ranura de tiempo. Todas las tramas tienen la longitud máxima permitida.

5

LA CAPA DE RED

La capa de red se encarga de llevar los paquetes desde el origen hasta el destino. Llegar al destino puede requerir muchos saltos por enrutadores intermedios. Esta función ciertamente contrasta con la de la capa de enlace de datos, que sólo tiene la meta modesta de mover tramas de un extremo del cable al otro. Por lo tanto, la capa de red es la capa más baja que maneja la transmisión de extremo a extremo.

Para lograr su cometido, la capa de red debe conocer la topología de la subred de comunicación (es decir, el grupo de enrutadores) y elegir las rutas adecuadas a través de ella; también debe tener cuidado al escoger las rutas para no sobrecargar algunas de las líneas de comunicación y los enrutadores y dejar inactivos a otros. Por último, cuando el origen y el destino están en redes diferentes, ocurren nuevos problemas. La capa de red es la encargada de solucionarlos. En este capítulo estudiaremos todos estos temas y los ilustraremos principalmente valiéndonos de Internet y de su protocolo de capa de red, IP, aunque también veremos las redes inalámbricas.

5.1 ASPECTOS DE DISEÑO DE LA CAPA DE RED

En las siguientes secciones presentaremos una introducción a algunos de los problemas que deben enfrentar los diseñadores de la capa de red. Estos temas incluyen el servicio proporcionado a la capa de transporte y el diseño interno de la subred.

5.1.1 Conmutación de paquetes de almacenamiento y reenvío

Antes de iniciar una explicación sobre los detalles de la capa de red, probablemente valga la pena volver a exponer el contexto en el que operan los protocolos de esta capa. En la figura 5-1 se muestra dicho contexto. Los componentes principales del sistema son el equipo de la empresa portadora (enrutadores conectados mediante líneas de transmisión), que se muestra dentro del óvalo sombreado, y el equipo del cliente, que se muestra fuera del óvalo. El *host H1* está conectado de manera directa al enrutador de una empresa portadora, *A*, mediante una línea alquilada. En contraste, *H2* se encuentra en una LAN con un enrutador, *F*, el cual es propiedad de un cliente, quien lo maneja. Este enrutador también tiene una línea alquilada hacia el equipo de la empresa portadora. Mostramos *F* fuera del óvalo porque no pertenece a la empresa portadora, pero en términos de construcción, software y protocolos, tal vez no sea diferente de los enrutadores de la empresa portadora. Si bien es debatible el hecho de que este enrutador pertenezca a la subred, para los propósitos de este capítulo, los enrutadores del cliente son considerados como parte de la subred porque se valen de los mismos algoritmos que los enrutadores de la empresa portadora (y nuestro interés principal aquí son los algoritmos).

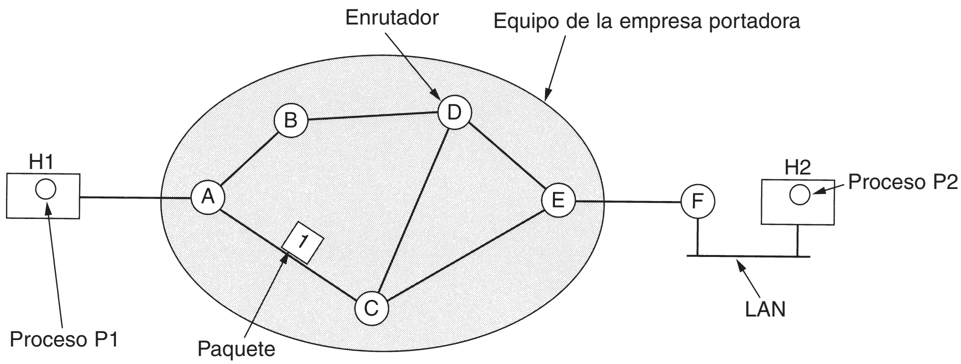


Figura 5-1. El entorno de los protocolos de la capa de red.

Este equipo se utiliza como sigue. Un *host* transmite al enrutador más cercano un paquete que tiene por enviar, ya sea en su propia LAN o a través de un enlace punto a punto con la empresa portadora. El paquete se almacena ahí hasta que haya llegado por completo, a fin de que la suma de verificación pueda comprobarse. Después se reenvía al siguiente enrutador de la ruta hasta que llegue al *host* de destino, donde se entrega. Este mecanismo se conoce como conmutación de paquetes de almacenamiento y reenvío, como vimos en capítulos anteriores.

5.1.2 Servicios proporcionados a la capa de transporte

La capa de red proporciona servicios a la capa de transporte en la interfaz capa de red/capa de transporte. Una pregunta importante es qué tipo de servicios proporciona la capa de red a la capa de transporte. Los servicios de la capa de red se han diseñado con los siguientes objetivos en mente.

1. Los servicios deben ser independientes de la tecnología del enrutador.
2. La capa de transporte debe estar aislada de la cantidad, tipo y topología de los enrutadores presentes.
3. Las direcciones de red disponibles para la capa de transporte deben seguir un plan de numeración uniforme, aun a través de varias LANs y WANs.

Dadas estas metas, los diseñadores de la capa de red tienen mucha libertad para escribir especificaciones detalladas de los servicios que se ofrecerán a la capa de transporte. Con frecuencia esta libertad degenera en una batalla campal entre dos bandos en conflicto. La discusión se centra en determinar si la capa de red debe proporcionar servicio orientado o no orientado a la conexión.

Un bando (representado por la comunidad de Internet) alega que la tarea del enrutador es mover bits de un lado a otro, y nada más. Desde su punto de vista (basado en casi 30 años de experiencia con una red de computadoras real y operativa), la subred es inherentemente inestable, sin importar su diseño. Por lo tanto, los *hosts* deben aceptar este hecho y efectuar ellos mismos el control de errores (es decir, detección y corrección de errores) y el control de flujo.

Este punto de vista conduce directamente a la conclusión de que el servicio de red no debe ser orientado a la conexión, y debe contar tan sólo con las primitivas SEND PACKET y RECEIVE PACKET. En particular, no debe efectuarse ningún ordenamiento de paquetes ni control de flujo, pues de todos modos los *hosts* lo van a efectuar y probablemente se ganaría poco haciéndolo dos veces. Además, cada paquete debe llevar la dirección de destino completa, porque cada paquete enviado se transporta de manera independiente de sus antecesores, si los hay.

El otro bando (representado por las compañías telefónicas) argumenta que la subred debe proporcionar un servicio confiable, orientado a la conexión. Afirman que una buena guía son 100 años de experiencia exitosa del sistema telefónico mundial. Desde este punto de vista, la calidad del servicio es el factor dominante, y sin conexiones en la subred, tal calidad es muy difícil de alcanzar, especialmente para el tráfico de tiempo real como la voz y el vídeo.

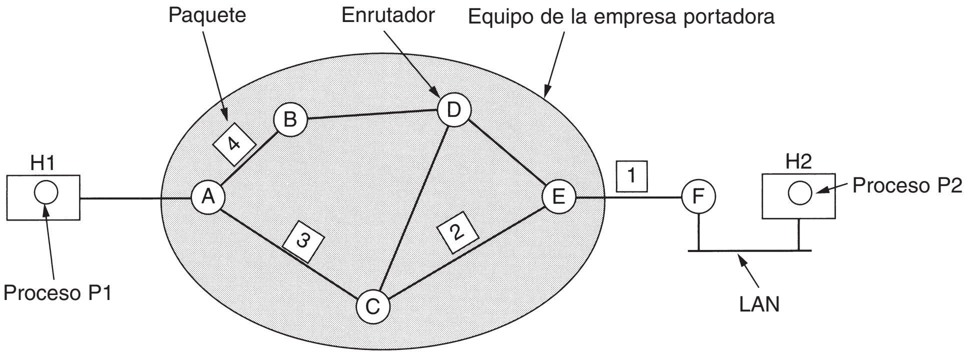
Estas dos posturas se ejemplifican mejor con Internet y ATM. Internet ofrece servicio de capa de red no orientado a la conexión; las redes ATM ofrecen servicio de capa de red orientado a la conexión. Sin embargo, es interesante hacer notar que conforme las garantías de calidad del servicio se están volviendo más y más importantes, Internet está evolucionando. En particular, está empezando a adquirir propiedades que normalmente se asocian con el servicio orientado a la conexión, como veremos más adelante. En el capítulo 4 dimos un breve indicio de esta evolución en nuestro estudio sobre las VLANs.

5.1.3 Implementación del servicio no orientado a la conexión

Puesto que ya vimos las dos clases de servicios que la capa de red puede proporcionar a sus usuarios, es tiempo de analizar la manera en que funciona internamente esta capa. Se pueden realizar dos formas de organización distintas, dependiendo del tipo de servicio que se ofrezca. Si se ofrece el servicio no orientado a la conexión, los paquetes se colocan individualmente en la subred y se enrutan de manera independiente. No se necesita una configuración avanzada. En este

contexto, por lo general los paquetes se conocen como **datagramas** (en analogía con los telegramas) y la subred se conoce como **subred de datagramas.** Si se utiliza el servicio orientado a la conexión, antes de poder enviar cualquier paquete de datos, es necesario establecer una ruta del enrutador de origen al de destino. Esta conexión se conoce como **CV** (**circuito virtual**), en analogía con los circuitos físicos establecidos por el sistema telefónico, y la subred se conoce como **subred de circuitos virtuales.** En esta sección examinaremos las subredes de datagramas; en la siguiente analizaremos las subredes de circuitos virtuales.

A continuación veamos cómo funciona una subred de datagramas. Suponga que el proceso *P1* de la figura 5-2 tiene un mensaje largo para *P2*. Dicho proceso entrega el mensaje a la capa de transporte y le indica a ésta que lo envíe al proceso *P2* que se encuentra en el *host H2*. El código de la capa de transporte se ejecuta en *H1*, por lo general dentro del sistema operativo. Dicho código agrega un encabezado de transporte al mensaje y entrega el resultado a la capa de red, quizá otro procedimiento dentro del sistema operativo.



Destino	Tabla de A inicialmente (Línea)	Tabla de A posteriormente (Línea)	Tabla de C (Línea)	Tabla de E (Línea)
A	–	–	A	C
B	B	B	A	D
C	C	C	–	C
D	B	B	D	D
E	C	B	E	–
F	C	B	E	F

Figura 5-2. Enrutamiento dentro de una subred de datagramas.

Supongamos que el mensaje es cuatro veces más largo que el tamaño máximo de paquete, por lo que la capa de red tiene que dividirlo en cuatro paquetes, 1, 2, 3 y 4, y envía cada uno de ellos a la vez al enrutador *A* mediante algún protocolo punto a punto; por ejemplo, PPP. En este momento entra en acción la empresa portadora. Cada enrutador tiene una tabla interna que le indica a dónde enviar paquetes para cada destino posible. Cada entrada de tabla es un par que consiste en un

destino y la línea de salida que se utilizará para ese destino. Sólo se pueden utilizar líneas conectadas directamente. Por ejemplo, en la figura 5-2, *A* sólo tiene dos líneas de salida —a *B* y *C*—, por lo que cada paquete entrante debe enviarse a uno de estos enrutadores, incluso si el destino final es algún otro enrutador. En la figura, la tabla de enrutamiento inicial de *A* se muestra abajo de la leyenda "inicialmente".

Conforme los paquetes 1, 2 y 3 llegaron a *A*, se almacenaron unos momentos (para comprobar sus sumas de verificación). Después cada uno se reenvió a *C* de acuerdo con la tabla de *A*. Posteriormente, el paquete 1 se reenvió a *E* y después a *F*. Cuando llegó a *F*, se encapsuló en una trama de capa de enlace de datos y se envió a *H2* a través de la LAN. Los paquetes 2 y 3 siguieron la misma ruta.

Sin embargo, pasó algo diferente con el paquete 4. Cuando llegó a *A*, se envió al enrutador *B*, aunque también estaba destinado a *F*. Por alguna razón, *A* decidió enviar el paquete 4 por una ruta diferente a la de los primeros tres paquetes. Tal vez se enteró de que había alguna congestión de tráfico en alguna parte de la ruta *ACE* y actualizó su tabla de enrutamiento, como se muestra bajo la leyenda "posteriormente". El algoritmo que maneja las tablas y que realiza las decisiones de enrutamiento se conoce como **algoritmo de enrutamiento.** Los algoritmos de enrutamiento son uno de los principales temas que estudiaremos en este capítulo.

5.1.4 Implementación del servicio orientado a la conexión

Para servicio orientado a la conexión necesitamos una subred de circuitos virtuales. Veamos cómo funciona. El propósito de los circuitos virtuales es evitar la necesidad de elegir una nueva ruta para cada paquete enviado, como en la figura 5-2. En su lugar, cuando se establece una conexión, se elige una ruta de la máquina de origen a la de destino como parte de la configuración de conexión y se almacena en tablas dentro de los enrutadores. Esa ruta se utiliza para todo el tráfico que fluye a través de la conexión, exactamente de la misma forma en que funciona el sistema telefónico. Cuando se libera la conexión, también se termina el circuito virtual. Con el servicio orientado a la conexión, cada paquete lleva un identificador que indica a cuál circuito virtual pertenece.

Como ejemplo, considere la situación que se muestra en la figura 5-3. En ésta, el *host H1* ha establecido una conexión 1 con el *host H2*. Se recuerda como la primera entrada de cada una de las tablas de enrutamiento. La primera línea de la tabla *A* indica que si un paquete tiene el identificador de conexión 1 viene de *H1*, se enviará al enrutador *C* y se le dará el identificador de conexión 1. De manera similar, la primera entrada en *C* enruta el paquete a *E*, también con el identificador de conexión 1.

Ahora consideremos lo que sucede si *H3* también desea establecer una conexión con *H2*. Elige el identificador de conexión 1 (debido a que está iniciando la conexión y a que ésta es su única conexión) y le indica a la subred que establezca el circuito virtual. Esto nos lleva a la segunda fila de las tablas. Observe que aquí surge un problema debido a que aunque *A* sí puede saber con facilidad cuáles paquetes de conexión 1 provienen de *H1* y cuáles provienen de *H3*, *C* no puede hacerlo. Por esta razón, *A* asigna un identificador de conexión diferente al tráfico saliente para la segunda conexión. Con el propósito de evitar conflictos de este tipo, los enrutadores requieren

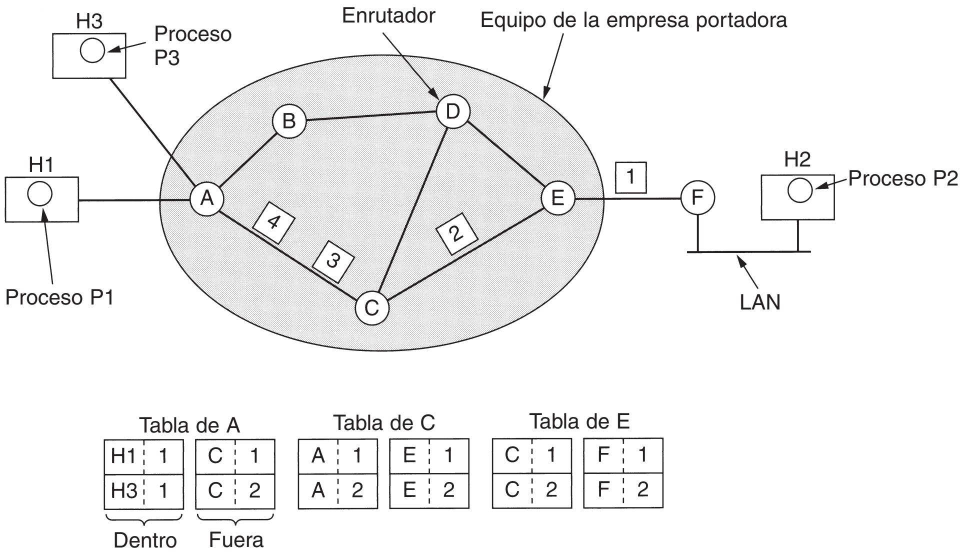


Figura 5-3. Enrutamiento dentro de una subred de circuitos virtuales.

la capacidad de reemplazar identificadores de conexión en los paquetes salientes. En algunos contextos a esto se le conoce como conmutación de etiquetas.

5.1.5 Comparación entre las subredes de circuitos virtuales y las de datagramas

Tanto los circuitos virtuales como los datagramas tienen sus seguidores y sus detractores. Ahora intentaremos resumir los argumentos de ambos bandos. Los aspectos principales se listan en la figura 5-4, aunque los puristas probablemente podrán encontrar ejemplos contrarios para todo lo indicado en la figura.

Dentro de la subred hay varios pros y contras entre los circuitos virtuales y los datagramas. Uno de ellos tiene que ver con el espacio de memoria del enrutador y el ancho de banda. Los circuitos virtuales permiten que los paquetes contengan números de circuito en lugar de direcciones de destino completas. Si los paquetes suelen ser bastante cortos, una dirección de destino completa en cada paquete puede representar una sobrecarga significativa y, por lo tanto, ancho de banda desperdiciado. El precio que se paga por el uso interno de circuitos virtuales es el espacio de tabla en los enrutadores. La mejor elección desde el punto de vista económico depende del costo relativo entre los circuitos de comunicación y la memoria de los enrutadores.

Otro punto por considerar es el del tiempo de configuración contra el tiempo de análisis de la dirección. El uso de circuitos virtuales requiere una fase de configuración, que consume tiempo y recursos. Sin embargo, determinar lo que hay que hacer con un paquete de datos en una subred de

Asunto	Subred de datagramas	Subred de circuitos virtuales
Configuración del circuito	No necesaria	Requerida
Direccionamiento	Cada paquete contiene la dirección de origen y de destino	Cada paquete contiene un número de CV corto
Información de estado	Los enrutadores no contienen información de estado de las conexiones	Cada CV requiere espacio de tabla del enrutador por conexión
Enrutamiento	Cada paquete se enruta de manera independiente	Ruta escogida cuando se establece el CV; todos los paquetes siguen esta ruta
Efecto de fallas del enrutador	Ninguno, excepto para paquetes perdidos durante una caída	Terminan todos los CVs que pasan a través del enrutador
Calidad del servicio	Difícil	Fácil si se pueden asignar suficientes recursos por adelantado para cada CV
Control de congestión	Difícil	Fácil si pueden asignarse por adelantado suficientes recursos a cada CV

Figura 5-4. Comparación de las subredes de datagramas y de circuitos virtuales.

circuitos virtuales es fácil: el enrutador simplemente usa el número de circuito para buscar en una tabla y encontrar hacia dónde va el paquete. En una subred de datagramas se requiere un procedimiento más complicado para localizar el destino del paquete.

Otra cuestión es la cantidad requerida de espacio de tabla en la memoria del enrutador. Una subred de datagramas necesita tener una entrada para cada destino posible, mientras que una subred de circuitos virtuales sólo necesita una entrada por cada circuito virtual. Sin embargo, esta ventaja es engañosa debido a que los paquetes de configuración de conexión también tienen que enrutarse, y a que utilizan direcciones de destino, de la misma forma en que lo hacen los datagramas.

Los circuitos virtuales tienen algunas ventajas en cuanto a la calidad del servicio y a que evitan congestiones en la subred, pues los recursos (por ejemplo, búferes, ancho de banda y ciclos de CPU) pueden reservarse por adelantado al establecerse la conexión. Una vez que comienzan a llegar los paquetes, estarán ahí el ancho de banda y la capacidad de enrutamiento necesarios. En una subred de datagramas es más difícil evitar las congestiones.

En los sistemas de procesamiento de transacciones (por ejemplo, las tiendas que llaman para verificar pagos con tarjeta de crédito), la sobrecarga requerida para establecer y terminar un circuito virtual puede ocupar mucho más tiempo que el uso real del circuito. Si la mayor parte del tráfico esperado es de este tipo, el uso de circuitos virtuales dentro de la subred tiene poco sentido. Por otra parte, aquí pueden ser de utilidad los circuitos virtuales permanentes, establecidos manualmente y con duración de meses o años.

Los circuitos virtuales también tienen un problema de vulnerabilidad. Si se cae un enrutador y se pierde su memoria, todos los circuitos virtuales que pasan por él tendrán que abortarse,

aunque se recupere un segundo después. Por el contrario, si se cae un enrutador de datagramas, sólo sufrirán los usuarios cuyos paquetes estaban encolados en el enrutador en el momento de la falla y, dependiendo de si ya se había confirmado o no su recepción, tal vez ni siquiera todos ellos. La pérdida de una línea de comunicación es fatal para los circuitos virtuales que la usan, pero puede compensarse fácilmente cuando se usan datagramas. Éstos también permiten que los enrutadores equilibren el tráfico a través de la subred, ya que las rutas pueden cambiarse a lo largo de una secuencia larga de transmisiones de paquetes.

5.2 ALGORITMOS DE ENRUTAMIENTO

La función principal de la capa de red es enrutar paquetes de la máquina de origen a la de destino. En la mayoría de las subredes, los paquetes requerirán varios saltos para completar el viaje. La única excepción importante son las redes de difusión, pero aun aquí es importante el enrutamiento si el origen y el destino no están en la misma red. Los algoritmos que eligen las rutas y las estructuras de datos que usan constituyen un aspecto principal del diseño de la capa de red.

El **algoritmo de enrutamiento** es aquella parte del software de la capa de red encargada de decidir la línea de salida por la que se transmitirá un paquete de entrada. Si la subred usa datagramas de manera interna, esta decisión debe tomarse cada vez que llega un paquete de datos, dado que la mejor ruta podría haber cambiado desde la última vez. Si la subred usa circuitos virtuales internamente, las decisiones de enrutamiento se toman sólo al establecerse un circuito virtual nuevo. En lo sucesivo, los paquetes de datos simplemente siguen la ruta previamente establecida. Este último caso a veces se llama **enrutamiento de sesión,** dado que una ruta permanece vigente durante toda la sesión de usuario (por ejemplo, durante una sesión desde una terminal, o durante una transferencia de archivos).

Algunas veces es útil distinguir entre el enrutamiento, que es el proceso consistente en tomar la decisión de cuáles rutas utilizar, y el reenvío, que consiste en la acción que se toma cuando llega un paquete. Se puede considerar que un enrutador realiza dos procesos internos. Uno de ellos maneja cada paquete conforme llega, buscando en las tablas de enrutamiento la línea de salida por la cual se enviará. Este proceso se conoce como **reenvío.** El otro proceso es responsable de llenar y actualizar las tablas de enrutamiento. Es ahí donde entra en acción el algoritmo de enrutamiento.

Sin importar si las rutas para cada paquete se eligen de manera independiente o sólo cuando se establecen nuevas conexiones, hay ciertas propiedades que todo algoritmo de enrutamiento debe poseer: exactitud, sencillez, robustez, estabilidad, equidad y optimización. La exactitud y la sencillez apenas requieren comentarios, pero la necesidad de robustez puede ser menos obvia a primera vista. Una vez que una red principal entra en operación, cabría esperar que funcionara continuamente durante años sin fallas a nivel de sistema. Durante ese periodo habrá fallas de hardware y de software de todo tipo. Los *hosts*, enrutadores y líneas fallarán en forma repetida y la topología cambiará muchas veces. El algoritmo de enrutamiento debe ser capaz de manejar los cambios de topología y tráfico sin requerir el aborto de todas las actividades en todos los *hosts* y el reinicio de la red con cada caída de un enrutador.

La estabilidad también es una meta importante del algoritmo de enrutamiento. Existen algoritmos de enrutamiento que nunca alcanzan el equilibrio, sin importar el tiempo que permanezcan operativos. Un algoritmo estable alcanza el equilibrio y lo conserva. La equidad y la optimización pueden parecer algo obvias (ciertamente nadie se opondrá a ellas), pero resulta que con frecuencia son metas contradictorias. En la figura 5-5 se muestra un ejemplo sencillo de este conflicto. Suponga que hay suficiente tráfico entre A y A', entre B y B' y entre C y C' para saturar los enlaces horizontales. A fin de aumentar al máximo el flujo total, el tráfico de X a X' debe suspenderse por completo. Por desgracia, X y X' podrían inconformarse. Evidentemente se requiere un punto medio entre la eficiencia global y la equidad hacia las conexiones individuales.

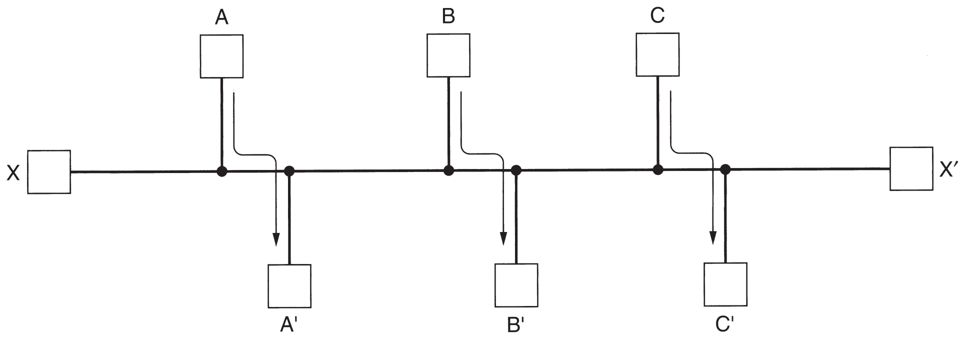


Figura 5-5. El conflicto entre equidad y optimización.

Antes de que podamos siquiera intentar encontrar el punto medio entre la equidad y la optimización, debemos decidir qué es lo que buscamos optimizar. Un candidato obvio es la minimización del retardo medio de los paquetes, pero también lo es el aumento al máximo de la velocidad real de transporte de la red. Además, estas dos metas también están en conflicto, ya que la operación de cualquier sistema de colas cerca de su capacidad máxima implica un retardo de encolamiento grande. Como término medio, muchas redes intentan minimizar el número de saltos que tiene que dar un paquete, puesto que la reducción de la cantidad de saltos reduce el retardo y también el consumo de ancho de banda, lo que a su vez mejora la velocidad real de transporte.

Los algoritmos de enrutamiento pueden agruparse en dos clases principales: no adaptativos y adaptativos. Los **algoritmos no adaptativos** no basan sus decisiones de enrutamiento en mediciones o estimaciones del tráfico y la topología actuales. En cambio, la decisión de qué ruta se usará para llegar de I a J (para todas las I y J) se toma por adelantado, fuera de línea, y se carga en los enrutadores al arrancar la red. Este procedimiento se conoce como **enrutamiento estático.**

En contraste, los **algoritmos adaptativos** cambian sus decisiones de enrutamiento para reflejar los cambios de topología y, por lo general también el tráfico. Los algoritmos adaptativos difieren en el lugar de donde obtienen su información (por ejemplo, localmente, de los enrutadores adyacentes o de todos los enrutadores), el momento de cambio de sus rutas (por ejemplo, cada ΔT segundos, cuando cambia la carga o cuando cambia la topología) y la métrica usada para la optimización (por ejemplo, distancia, número de saltos o tiempo estimado de tránsito). En las siguientes

secciones estudiaremos una variedad de algoritmos de enrutamiento, tanto estáticos como dinámicos.

5.2.1 Principio de optimización

Antes de entrar en algoritmos específicos, puede ser útil señalar que es posible hacer un postulado general sobre las rutas óptimas sin importar la topología o el tráfico de la red. Este postulado se conoce como **principio de optimización,** y establece que si el enrutador *J* está en ruta óptima del enrutador *I* al enrutador *K*, entonces la ruta óptima de *J* a *K* también está en la misma ruta. Para ver esto, llamemos r_1 a la parte de la ruta de *I* a Jr_1 y r_2 al resto de la ruta. Si existiera una ruta mejor que r_2 entre *J* y *K*, podría conectarse con r_1 para mejorar la ruta entre *I* y *K*, contradiciendo nuestra aseveración de que r_1r_2 es óptima.

Como consecuencia directa del principio de optimización, podemos ver que el grupo de rutas óptimas de todos los orígenes a un destino dado forman un árbol con raíz en el destino. Tal árbol se conoce como **árbol sumidero** (o árbol divergente) y se ilustra en la figura 5-6, donde la métrica de distancia es el número de saltos. Observe que un árbol sumidero no necesariamente es único; pueden existir otros árboles con las mismas longitudes de rutas. La meta de todos los algoritmos de enrutamiento es descubrir y utilizar los árboles sumideros de todos los enrutadores.

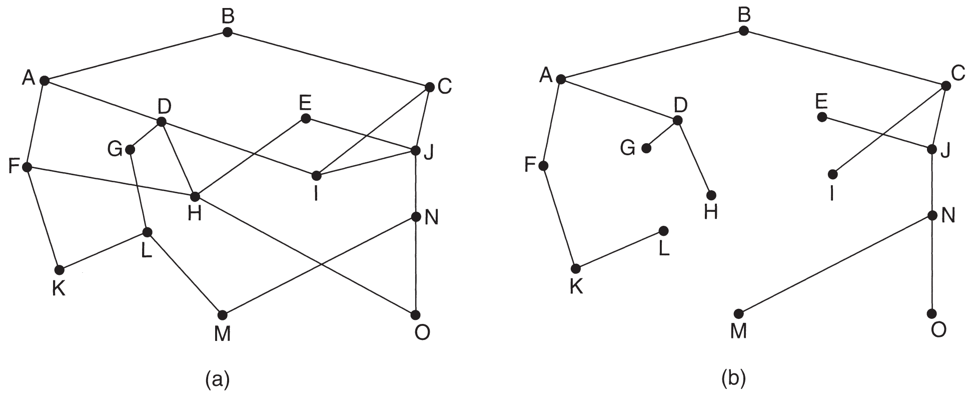


Figura 5-6. (a) Una subred. (b) Árbol sumidero para el enrutador *B*.

Puesto que un árbol sumidero ciertamente es un árbol, no contiene ciclos, por lo que cada paquete será entregado en un número de saltos finito y limitado. En la práctica, la vida no es tan fácil. Los enlaces y los enrutadores pueden caerse y reactivarse durante la operación, por lo que los diferentes enrutadores pueden tener ideas distintas sobre la topología actual. Además hemos evadido calladamente la cuestión de si cada enrutador tiene que adquirir de manera individual la información en la cual basa su cálculo del árbol sumidero, o si esta información se obtiene por otros

medios. Regresaremos a estos asuntos pronto. Con todo, el principio de optimización y el árbol sumidero proporcionan parámetros contra los que se pueden medir otros algoritmos de enrutamiento.

5.2.2 Enrutamiento por la ruta más corta

Comencemos nuestro estudio de los algoritmos de enrutamiento con una técnica de amplio uso en muchas formas, porque es sencilla y fácil de entender. La idea es armar un grafo de la subred, en el que cada nodo representa un enrutador y cada arco del grafo una línea de comunicación (con frecuencia llamada enlace). Para elegir una ruta entre un par dado de enrutadores, el algoritmo simplemente encuentra en el grafo la ruta más corta entre ellos.

El concepto de **ruta más corta** merece una explicación. Una manera de medir la longitud de una ruta es por la cantidad de saltos. Usando esta métrica, las rutas *ABC* y *ABE* de la figura 5-7 tienen la misma longitud. Otra métrica es la distancia geográfica en kilómetros, en cuyo caso *ABC* es claramente mucho mayor que *ABE* (suponiendo que la figura está dibujada a escala).

Sin embargo, también son posibles muchas otras métricas además de los saltos y la distancia física. Por ejemplo, cada arco podría etiquetarse con el retardo medio de encolamiento y transmisión de un paquete de prueba estándar, determinado por series de prueba cada hora. Con estas etiquetas en el grafo, la ruta más corta es la más rápida, en lugar de la ruta con menos arcos o kilómetros.

En el caso más general, las etiquetas de los arcos podrían calcularse como una función de la distancia, ancho de banda, tráfico medio, costo de comunicación, longitud media de las colas, retardo medio y otros factores. Cambiando la función de ponderación, el algoritmo calcularía la ruta "más corta" de acuerdo con cualquiera de varios criterios, o una combinación de ellos.

Se conocen varios algoritmos de cálculo de la ruta más corta entre dos nodos de un grafo. Éste se debe a Dijkstra (1959). Cada nodo se etiqueta (entre paréntesis) con su distancia al nodo de origen a través de la mejor ruta conocida. Inicialmente no se conocen rutas, por lo que todos los nodos tienen la etiqueta infinito. A medida que avanza el algoritmo y se encuentran rutas, las etiquetas pueden cambiar, reflejando mejores rutas. Una etiqueta puede ser tentativa o permanente. Inicialmente todas las etiquetas son tentativas. Una vez que se descubre que una etiqueta representa la ruta más corta posible del origen a ese nodo, se vuelve permanente y no cambia más.

Para ilustrar el funcionamiento del algoritmo de etiquetado, observe el grafo ponderado no dirigido de la figura 5-7(a), donde las ponderaciones representan, por ejemplo, distancias. Queremos encontrar la ruta más corta posible de *A* a *D*. Comenzamos por marcar como permanente el nodo *A*, indicado por un círculo rellenado. Después examinamos, por turno, cada uno de los nodos adyacentes a *A* (el nodo de trabajo), reetiquetando cada uno con la distancia desde *A*. Cada vez que reetiquetamos un nodo, también lo reetiquetamos con el nodo desde el que se hizo la prueba, para poder reconstruir más tarde la ruta final. Una vez que terminamos de examinar cada uno de los nodos adyacentes a *A*, examinamos todos los nodos etiquetados tentativamente en el grafo

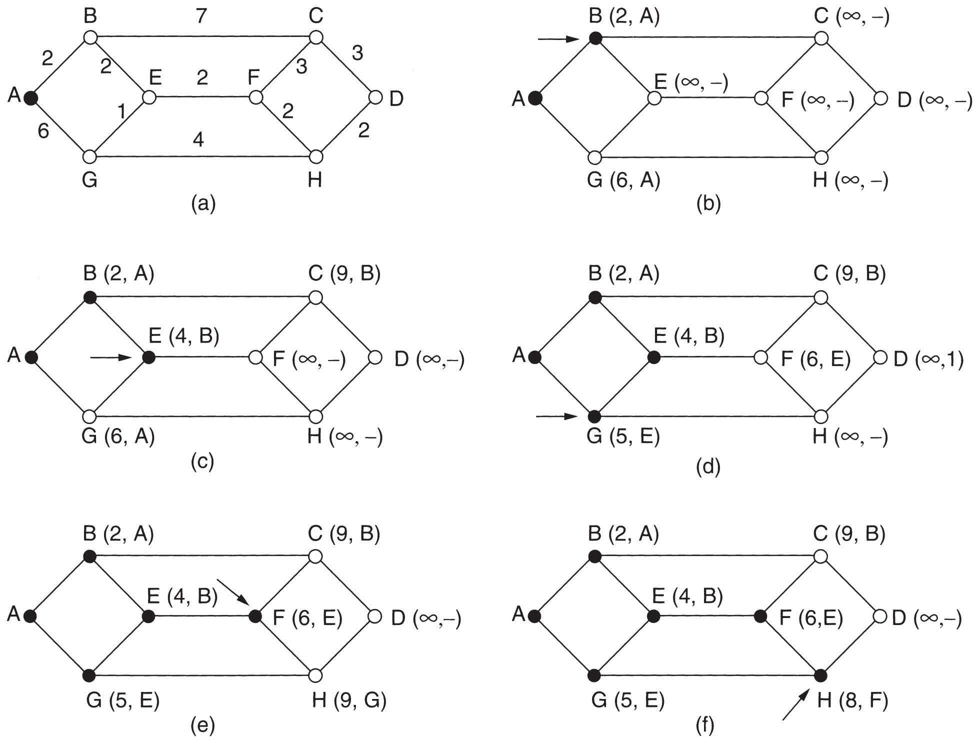


Figura 5-7. Los primeros cinco pasos del cálculo de la ruta más corta de *A* a *D*. Las flechas indican el nodo de trabajo.

completo y hacemos permanente el de la etiqueta más pequeña, como se muestra en la figura 5-7(b). Éste se convierte en el nuevo nodo de trabajo.

Ahora comenzamos por *B*, y examinamos todos los nodos adyacentes a él. Si la suma de la etiqueta de *B* y la distancia desde *B* al nodo en consideración es menor que la etiqueta de ese nodo, tenemos una ruta más corta, por lo que reetiquetamos ese nodo.

Tras inspeccionar todos los nodos adyacentes al nodo de trabajo y cambiar las etiquetas tentativas (de ser posible), se busca en el grafo completo el nodo etiquetado tentativamente con el menor valor. Este nodo se hace permanente y se convierte en el nodo de trabajo para la siguiente ronda. En la figura 5-7 se muestran los primeros cinco pasos del algoritmo.

Para ver por qué funciona el algoritmo, vea la figura 5-7(c). En ese punto acabamos de hacer permanente a *E*. Suponga que hubiera una ruta más corta que *ABE*, digamos *AXYZE*. Hay dos posibilidades: el nodo *Z* ya se hizo permanente, o no se ha hecho permanente. Si ya es permanente, entonces *E* ya se probó (en la ronda que siguió a aquella en la que se hizo permanente *Z*), por lo que la ruta *AXYZE* no ha escapado a nuestra atención y, por lo tanto, no puede ser una ruta más corta.

Ahora considere el caso en el que *Z* aún está etiquetado tentativamente. O bien la etiqueta de *Z* es mayor o igual que la de *E*, en cuyo caso *AXYZE* no puede ser una ruta más corta que *ABE*, o es menor que la de *E*, en cuyo caso *Z*, y no *E*, se volverá permanente primero, lo que permitirá que *E* se pruebe desde *Z*.

Este algoritmo se da en la figura 5-8. Las variables globales *n* y *dist* describen el grafo y son inicializadas antes de que se llame a *shortest_path*. La única diferencia entre el programa y el algoritmo antes descrito es que, en la figura 5-8, calculamos la ruta más corta posible comenzando por el nodo terminal, *t*, en lugar de en el nodo de origen, *s*. Dado que la ruta más corta posible desde *t* a *s* en un grafo no dirigido es igual a la ruta más corta de *s* a *t*, no importa el extremo por el que comencemos (a menos que haya varias rutas más cortas posibles, en cuyo caso la inversión de la búsqueda podría descubrir una distinta). La razón de una búsqueda hacia atrás es que cada nodo está etiquetado con su antecesor, en lugar de con su sucesor. Al copiar la ruta final en la variable de salida, *path*, la ruta de salida se invierte. Al invertir la búsqueda, ambos efectos se cancelan y la respuesta se produce en el orden correcto.

5.2.3 Inundación

Otro algoritmo estático es la **inundación,** en la que cada paquete de entrada se envía por cada una de las líneas de salida, excepto aquella por la que llegó. La inundación evidentemente genera grandes cantidades de paquetes duplicados; de hecho, una cantidad infinita a menos que se tomen algunas medidas para limitar el proceso. Una de estas medidas es integrar un contador de saltos en el encabezado de cada paquete, que disminuya con cada salto, y el paquete se descarte cuando el contador llegue a cero. Lo ideal es inicializar el contador de saltos a la longitud de la ruta entre el origen y el destino. Si el emisor desconoce el tamaño de la ruta, puede inicializar el contador al peor caso, es decir, el diámetro total de la subred.

Una técnica alterna para ponerle diques a la inundación es llevar un registro de los paquetes difundidos, para evitar enviarlos una segunda vez. Una manera de lograr este propósito es hacer que el enrutador de origen ponga un número de secuencia en cada paquete que recibe de sus *hosts*. Cada enrutador necesita una lista por cada enrutador de origen que indique los números de secuencia originados en ese enrutador que ya ha visto. Si un paquete de entrada está en la lista, no se difunde.

Para evitar que la lista crezca sin límites, cada lista debe incluir un contador, *k*, que indique que todos los números de secuencia hasta *k* ya han sido vistos. Cuando llega un paquete, es fácil comprobar si es un duplicado; de ser así, se descarta. Es más, no se necesita la lista completa por debajo de *k*, pues *k* la resume efectivamente.

Una variación de la inundación, un poco más práctica, es la **inundación selectiva.** En este algoritmo, los enrutadores no envían cada paquete de entrada por todas las líneas, sino sólo por aquellas que van aproximadamente en la dirección correcta. Por lo general, no tiene mucho caso enviar un paquete dirigido al oeste a través de una línea dirigida al este, a menos que la topología sea extremadamente peculiar y que el enrutador esté seguro de este hecho.

```
#define MAX_NODES 1024                    /* número máximo de nodos */
#define INFINITY 1000000000               /* un número mayor que cualquier ruta
                                             máxima */
int n, dist[MAX_NODES][MAX_NODES];        /* dist[i][j] es la distancia entre
                                             i y j */

void shortest_path(int s, int t, int path[])
{ struct state {                          /* la ruta con la que se está
                                             trabajando */
     int predecessor;                     /* nodo previo */
     int length;                          /* longitud del origen a este nodo */
     enum {permanent, tentative} label;/* estado de la etiqueta */
 } state[MAX_NODES];

 int i, k, min;
 struct state *p;

 for (p = &state[0]; p < &state[n]; p++) {    /* estado de inicialización */
      p->predecessor = -1;
      p->length = INFINITY;
      p->label = tentative;
 }
 state[t].length = 0; state[t].label = permanent;
 k = t;                                   /* k es el nodo de trabajo inicial */
 do{                                      /* ¿hay una ruta mejor desde k? */
   for (i = 0; i < n; i++)                /* este grafo tiene n nodos */
         if (dist[k][i] != 0 && state[i].label == tentative) {
                if (state[k].length + dist[k][i] < state[i].length) {
                     state[i].predecessor = k;
                     state[i].length = state[k].length + dist[k][i];
                }
         }
   /* Encuentra el nodo etiquetado tentativamente con la etiqueta menor. */
   k = 0; min = INFINITY;
   for (i = 0; i < n; i++)
         if (state[i].label == tentative && state[i].length < min) {
               min = state[i].length;
               k = i;
         }
   state[k].label = permanent;
 } while (k != s);

 /* Copia la ruta en el arreglo de salida. */
 i = 0; k = s;
 do {path[i++] = k; k = state[k].predecessor; } while (k >= 0);
}
```

Figura 5-8. Algoritmo de Dijkstra para calcular la ruta más corta a través de un grafo.

La inundación no es práctica en la mayoría de las aplicaciones, pero tiene algunos usos. Por ejemplo, en aplicaciones militares, donde grandes cantidades de enrutadores pueden volar en pedazos en cualquier momento, es altamente deseable la excelente robustez de la inundación. En las aplicaciones distribuidas de bases de datos a veces es necesario actualizar concurrentemente todas las bases de datos, en cuyo caso la inundación puede ser útil. En las redes inalámbricas, algunas estaciones que se encuentren dentro del alcance de radio de una estación dada pueden recibir los mensajes que ésta trasmita, lo cual es, de hecho, inundación, y algunos algoritmos utilizan esta propiedad. Un cuarto posible uso de la inundación es como métrica contra la que pueden compararse otros algoritmos de enrutamiento. La inundación siempre escoge la ruta más corta posible, porque escoge en paralelo todas las rutas posibles. En consecuencia, ningún otro algoritmo puede producir un retardo más corto (si ignoramos la sobrecarga generada por el proceso de inundación mismo).

5.2.4 Enrutamiento por vector de distancia

Las redes modernas de computadoras por lo general utilizan algoritmos de enrutamiento dinámico en lugar de los estáticos antes descritos, pues los algoritmos estáticos no toman en cuenta la carga actual de la red. En particular, dos algoritmos dinámicos, el enrutamiento por vector de distancia y el enrutamiento por estado del enlace, son los más comunes. En esta sección veremos el primer algoritmo. En la siguiente estudiaremos el segundo.

Los algoritmos de **enrutamiento por vector de distancia** operan haciendo que cada enrutador mantenga una tabla (es decir, un vector) que da la mejor distancia conocida a cada destino y la línea que se puede usar para llegar ahí. Estas tablas se actualizan intercambiando información con los vecinos.

El algoritmo de enrutamiento por vector de distancia a veces recibe otros nombres, incluido el de algoritmo de enrutamiento **Bellman-Ford** distribuido y el de algoritmo **Ford-Fulkerson,** por los investigadores que los desarrollaron (Bellman, 1957, y Ford y Fulkerson, 1962). Éste fue el algoritmo original de enrutamiento de ARPANET y también se usó en Internet con el nombre RIP.

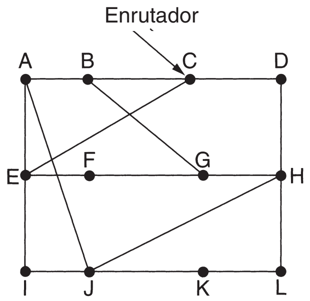
En el enrutamiento por vector de distancia, cada enrutador mantiene una tabla de enrutamiento indizada por, y conteniendo un registro de, cada enrutador de la subred. Esta entrada comprende dos partes: la línea preferida de salida hacia ese destino y una estimación del tiempo o distancia a ese destino. La métrica usada podría ser la cantidad de saltos, el retardo de tiempo en milisegundos, el número total de paquetes encolados a lo largo de la ruta, o algo parecido.

Se supone que el enrutador conoce la "distancia" a cada uno de sus vecinos. Si la métrica es de saltos, la distancia simplemente es un salto. Si la métrica es la longitud de la cola, el enrutador simplemente examina cada cola. Si la métrica es el retardo, el enrutador puede medirlo en forma directa con paquetes especiales de ECO que el receptor simplemente marca con la hora y lo regresa tan rápido como puede.

Por ejemplo, suponga que el retardo se usa como métrica y que el enrutador conoce el retardo a cada uno de sus vecinos. Una vez cada *T* mseg, cada enrutador envía a todos sus vecinos una

lista de sus retardos estimados a cada destino. También recibe una lista parecida de cada vecino. Imagine que una de estas tablas acaba de llegar del vecino *X*, siendo X_i la estimación de *X* respecto al tiempo que le toma llegar al enrutador *i*. Si el enrutador sabe que el retardo a *X* es de *m* mseg, también sabe que puede llegar al enrutador *i* a través de *X* en $X_i + m$ mseg. Efectuando este cálculo para cada vecino, un enrutador puede encontrar la estimación que parezca ser la mejor y usar esa estimación, así como la línea correspondiente, en su nueva tabla de enrutamiento. Observe que la vieja tabla de enrutamiento no se usa en este cálculo.

Este proceso de actualización se ilustra en la figura 5-9. En la parte (a) se muestra una subred. En las primeras cuatro columnas de la parte (b) aparecen los vectores de retardo recibidos de los vecinos del enrutador *J*. *A* indica tener un retardo de 12 mseg a *B*, un retardo de 25 mseg a *C*, un retardo de 40 mseg a *D*, etc. Suponga que *J* ha medido o estimado el retardo a sus vecinos *A*, *I*, *H* y *K* en 8, 10, 12 y 6 mseg, respectivamente.



(a)

	A	I	H	K	Nuevo retardo estimado desde J	Línea
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	Retardo JA es de 8	Retardo JI es de 10	Retardo JH es de 12	Retardo JK es de 6	Nueva tabla de enrutamiento para J	

Vectores recibidos de los cuatro vecinos de J

(b)

Figura 5-9. (a) Subred. (b) Entrada de A, I, H, K y la nueva tabla de enrutamiento de J.

Considere la manera en que *J* calcula su nueva ruta al enrutador *G*. Sabe que puede llegar a *A* en 8 mseg, y *A* indica ser capaz de llegar a *G* en 18 mseg, por lo que *J* sabe que puede contar con un retardo de 26 mseg a *G* si reenvía a través de *A* los paquetes destinados a *G*. Del mismo modo, *J* calcula el retardo a *G* a través de *I*, *H* y *K* en 41 (31 + 10), 18 (6 + 12) y 37 (31 + 6) mseg, respectivamente. El mejor de estos valores es el 18, por lo que escribe una entrada en su tabla de enrutamiento indicando que el retardo a *G* es de 18 mseg, y que la ruta que se utilizará es vía *H*. Se lleva a cabo el mismo cálculo para los demás destinos, y la nueva tabla de enrutamiento se muestra en la última columna de la figura.

El problema de la cuenta hasta infinito

El enrutamiento por vector de distancia funciona en teoría, pero tiene un problema serio en la práctica: aunque llega a la respuesta correcta, podría hacerlo lentamente. En particular, reacciona con rapidez a las buenas noticias, pero con lentitud ante las malas. Considere un enrutador cuya mejor ruta al destino *X* es larga. Si en el siguiente intercambio el vecino *A* informa repentinamente un retardo corto a *X*, el enrutador simplemente se conmuta a modo de usar la línea a *A* para enviar tráfico hasta *X*. En un intercambio de vectores, se procesan las buenas noticias.

Para ver la rapidez de propagación de las buenas noticias, considere la subred de cinco nodos (lineal) de la figura 5-10, en donde la métrica de retardo es el número de saltos. Suponga que *A* está desactivado inicialmente y que los otros enrutadores lo saben. En otras palabras, habrán registrado como infinito el retardo a *A*.

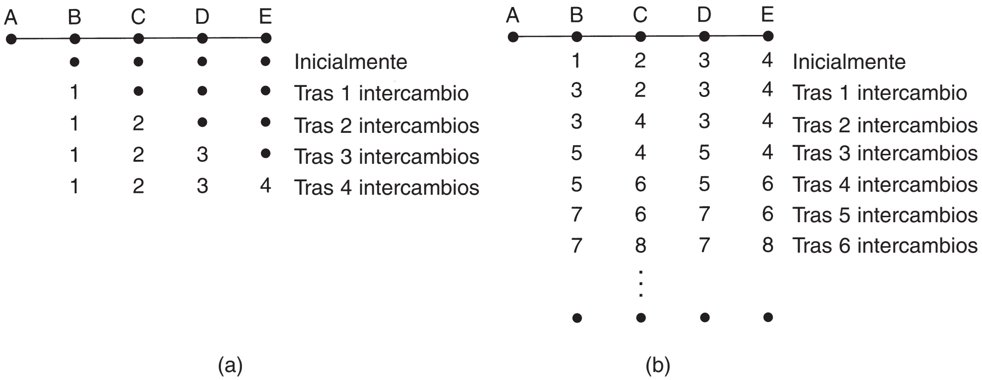


Figura 5-10. El problema de la cuenta hasta infinito.

Al activarse *A*, los demás enrutadores saben de él gracias a los intercambios de vectores. Por sencillez, supondremos que hay un gong gigantesco en algún lado, golpeando periódicamente para iniciar de manera simultánea un intercambio de vectores entre todos los enrutadores. En el momento del primer intercambio, *B* se entera de que su vecino de la izquierda tiene un retardo de 0 hacia *A*. *B* crea entonces una entrada en su tabla de enrutamiento, indicando que *A* está a un salto de distancia hacia la izquierda. Los demás enrutadores aún piensan que *A* está desactivado. En este punto, las entradas de la tabla de enrutamiento de *A* se muestran en la segunda fila de la figura 5-10(a). Durante el siguiente intercambio, *C* se entera de que *B* tiene una ruta a *A* de longitud 1, por lo que actualiza su tabla de enrutamiento para indicar una ruta de longitud 2, pero *D* y *E* no se enteran de las buenas nuevas sino hasta después. Como es evidente, las buenas noticias se difunden a razón de un salto por intercambio. En una subred cuya ruta mayor tiene una longitud de *N* saltos, en un lapso de *N* intercambios todo mundo sabrá sobre las líneas y enrutadores recientemente revividos.

Ahora consideremos la situación de la figura 5-10(b), en la que todas las líneas y enrutadores están activos inicialmente. Los enrutadores *B*, *C*, *D* y *E* tienen distancias a *A* de 1, 2, 3 y 4, respectivamente. De pronto, *A* se desactiva, o bien se corta la línea entre *A* y *B*, que de hecho es la misma cosa desde el punto de vista de *B*.

En el primer intercambio de paquetes, *B* no escucha nada de *A*. Afortunadamente, *C* dice: “No te preocupes. Tengo una ruta a *A* de longitud 2”. *B* no sabe que la ruta de *C* pasa a través de *B* mismo. Hasta donde *B* sabe, *C* puede tener 10 líneas, todas con rutas independientes a *A* de longitud 2. Como resultado, *B* ahora piensa que puede llegar a *A* por medio de *C*, con una longitud de ruta de 3. *D* y *E* no actualizan sus entradas para *A* en el primer intercambio.

En el segundo intercambio, *C* nota que cada uno de sus vecinos indica tener una ruta a *A* de longitud 3. *C* escoge una de ellas al azar y hace que su nueva distancia a *A* sea de 4, como se muestra en la tercera fila de la figura 5-10(b). Los intercambios subsecuentes producen la historia mostrada en el resto de la figura 5-10(b).

A partir de esta figura debe quedar clara la razón por la que las malas noticias viajan con lentitud: ningún enrutador jamás tiene un valor mayor en más de una unidad que el mínimo de todos sus vecinos. Gradualmente, todos los enrutadores elevan cuentas hacia el infinito, pero el número de intercambios requerido depende del valor numérico usado para el infinito. Por esta razón, es prudente hacer que el infinito sea igual a la ruta más larga, más 1. Si la métrica es el retardo de tiempo, no hay un límite superior bien definido, por lo que se necesita un valor alto para evitar que una ruta con un retardo grande sea tratada como si estuviera desactivada. Este problema se conoce como el problema de la **cuenta hasta el infinito,** lo cual no es del todo sorprendente. Se han realizado algunos intentos por resolverlo (como el horizonte dividido con rutas inalcanzables [*poisoned reverse*] en el RFC 1058), pero ninguno funciona bien en general. La esencia del problema consiste en que cuando *X* indica a *Y* que tiene una ruta en algún lugar, *Y* no tiene forma de saber si él mismo está en la ruta.

5.2.5 Enrutamiento por estado del enlace

El enrutamiento por vector de distancia se usó en ARPANET hasta 1979, cuando fue reemplazado por el enrutamiento por estado del enlace. Dos problemas principales causaron su desaparición. Primero, debido a que la métrica de retardo era la longitud de la cola, no tomaba en cuenta el ancho de banda al escoger rutas. Inicialmente, todas las líneas eran de 56 kbps, por lo que el ancho de banda no era importante, pero una vez que se modernizaron algunas líneas a 230 kbps y otras a 1.544 Mbps, el no tomar en cuenta el ancho de banda se volvió un problema importante. Por supuesto, habría sido posible cambiar la métrica de retardo para considerar el ancho de banda, pero también existía un segundo problema: que el algoritmo con frecuencia tardaba demasiado en converger (el problema de la cuenta hasta el infinito). Por estas razones, el algoritmo fue reemplazado por uno completamente nuevo, llamado **enrutamiento por estado del enlace.** Hoy en día se usan bastante algunas variantes del enrutamiento por estado del enlace.

El concepto en que se basa el enrutamiento por estado del enlace es sencillo y puede enunciarse en cinco partes. Cada enrutador debe:

1. Descubrir a sus vecinos y conocer sus direcciones de red.
2. Medir el retardo o costo para cada uno de sus vecinos.
3. Construir un paquete que indique todo lo que acaba de aprender.
4. Enviar este paquete a todos los demás enrutadores.
5. Calcular la ruta más corta a todos los demás enrutadores.

De hecho, toda la topología y todos los retardos se miden experimentalmente y se distribuyen a cada enrutador. Entonces puede usarse el algoritmo de Dijkstra para encontrar la ruta más corta a los demás enrutadores. A continuación veremos con mayor detalle estos cinco pasos.

Conocimiento de los vecinos

Cuando un enrutador se pone en funcionamiento, su primera tarea es averiguar quiénes son sus vecinos; esto lo realiza enviando un paquete HELLO especial a cada línea punto a punto. Se espera que el enrutador del otro extremo regrese una respuesta indicando quién es. Estos nombres deben ser globalmente únicos puesto que, cuando un enrutador distante escucha después que tres enrutadores están conectados a *F*, es indispensable que pueda determinar si los tres se refieren al mismo *F*.

Cuando se conectan dos o más enrutadores mediante una LAN, la situación es ligeramente más complicada. En la figura 5-11(a) se ilustra una LAN a la que están conectados directamente tres enrutadores, *A*, *C* y *F*. Cada uno de estos enrutadores está conectado a uno o más enrutadores adicionales.

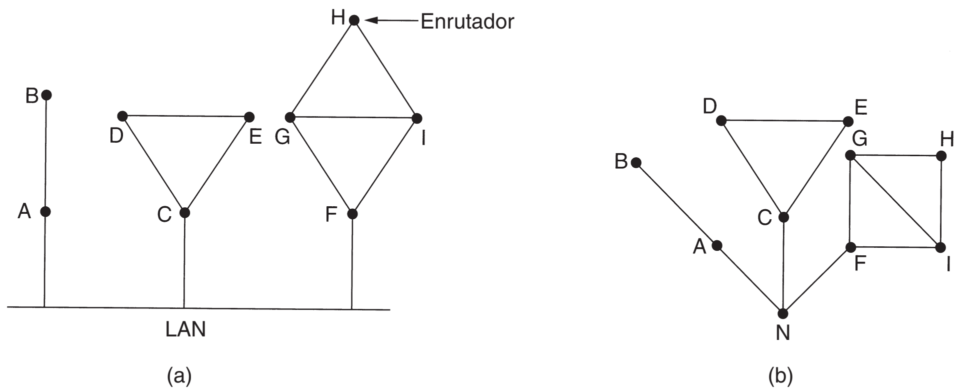


Figura 5-11. (a) Nueve enrutadores y una LAN. (b) Modelo de grafo de (a).

Una manera de modelar la LAN es considerarla como otro nodo, como se muestra en la figura 5-11(b). Aquí hemos introducido un nodo artificial nuevo, *N*, al que están conectados *A*, *C* y *F*. El hecho de que sea posible ir de *A* a *C* a través de la LAN se representa aquí mediante la ruta *ANC*.

Medición del costo de la línea

El algoritmo de enrutamiento por estado del enlace requiere que cada enrutador sepa, o cuando menos tenga una idea razonable, del retardo a cada uno de sus vecinos. La manera más directa de determinar este retardo es enviar un paquete ECHO especial a través de la línea y una vez que llegue al otro extremo, éste debe regresarlo inmediatamente. Si se mide el tiempo de ida y vuelta y se divide entre dos, el enrutador emisor puede tener una idea razonable del retardo. Para

obtener todavía mejores resultados, la prueba puede llevarse a cabo varias veces y usarse el promedio. Por supuesto que este método asume de manera implícita que los retardos son simétricos, lo cual no siempre es el caso.

Un aspecto interesante es si se debe tomar en cuenta la carga al medir el retardo. Para considerar la carga, el temporizador debe iniciarse cuando el paquete ECHO se ponga en la cola. Para ignorar la carga, el temporizador debe iniciarse cuando el paquete ECHO alcance el frente de la cola.

Pueden citarse argumentos a favor de ambos métodos. La inclusión de los retardos inducidos por el tráfico en las mediciones implica que cuando un enrutador puede escoger entre dos líneas con el mismo ancho de banda, una con carga alta continua y otra sin ella, considerará como ruta más corta la de la línea sin carga. Esta selección resultará en un mejor desempeño.

Desgraciadamente, también hay un argumento en contra de la inclusión de la carga en el cálculo del retardo. Considere la subred de la figura 5-12, dividida en dos partes, este y oeste, conectadas por dos líneas, *CF* y *EI*.

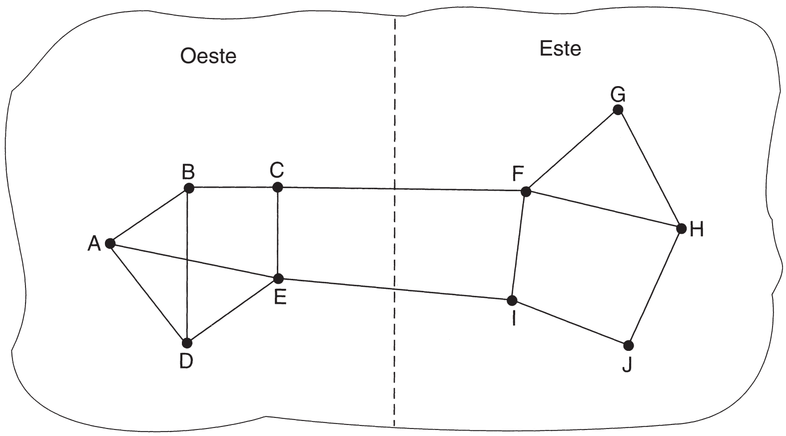


Figura 5-12. Subred en la que las partes este y oeste están conectadas por dos líneas.

Suponga que la mayor parte del tráfico entre el este y el oeste usa la línea *CF* y, como resultado, esta línea tiene tráfico alto con retardos grandes. La inclusión del retardo por encolamiento en el cálculo de la ruta más corta hará más atractiva a *EI*. Una vez instaladas las nuevas tablas de enrutamiento, la mayor parte del tráfico este-oeste pasará ahora por *EI*, sobrecargando esta línea. En consecuencia, en la siguiente actualización, *CF* aparecerá como la ruta más corta. Como resultado, las tablas de enrutamiento pueden oscilar sin control, lo que provocará un enrutamiento errático y muchos problemas potenciales. Si se ignora la carga y sólo se considera el ancho de banda, no ocurre este problema. De manera alterna, puede distribuirse la carga entre ambas líneas, pero esta solución no aprovecha al máximo la mejor ruta. No obstante, para evitar oscilaciones en la selección de la mejor ruta, podría ser adecuado dividir la carga entre múltiples líneas, con una fracción conocida de la carga viajando sobre cada una de ellas.

Construcción de los paquetes de estado del enlace

Una vez que se ha recabado la información necesaria para el intercambio, el siguiente paso es que cada enrutador construya un paquete que contenga todos los datos. El paquete comienza con la identidad del emisor, seguida de un número de secuencia, una edad (que se describirá después) y una lista de vecinos. Se da el retardo de vecino. En la figura 5-13(a) se da un ejemplo de subred, y los retardos se muestran como etiquetas en las líneas. En la figura 5-13(b) se muestran los paquetes de estado del enlace de los seis enrutadores.

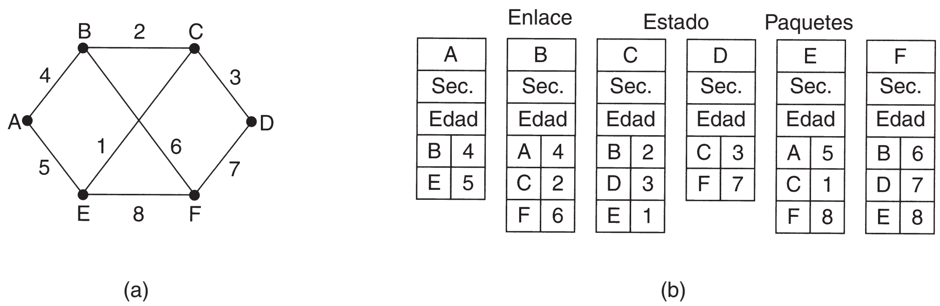


Figura 5-13. (a) Subred. (b) Paquetes de estado del enlace para esta subred.

Es fácil construir los paquetes de estado del enlace. La parte difícil es determinar cuándo construirlos. Una posibilidad es construirlos de manera periódica, es decir, a intervalos regulares. Otra posibilidad es construirlos cuando ocurra un evento significativo, como la caída o la reactivación de una línea o de un vecino, o el cambio apreciable de sus propiedades.

Distribución de los paquetes de estado del enlace

La parte más complicada del algoritmo es la distribución confiable de los paquetes de estado del enlace. A medida que se distribuyen e instalan los paquetes, los enrutadores que reciban los primeros cambiarán sus rutas. En consecuencia, los distintos enrutadores podrían estar usando versiones diferentes de la topología, lo que puede conducir a inconsistencias, ciclos, máquinas inalcanzables y otros problemas.

Primero describiremos el algoritmo básico de distribución y luego lo refinaremos. La idea fundamental es utilizar inundación para distribuir los paquetes de estado del enlace. A fin de mantener controlada la inundación, cada paquete contiene un número de secuencia que se incrementa con cada paquete nuevo enviado. Los enrutadores llevan el registro de todos los pares (enrutador de origen, secuencia) que ven. Cuando llega un paquete de estado del enlace, se verifica contra la lista de paquetes ya vistos. Si es nuevo, se reenvía a través de todas las líneas, excepto aquella por la que llegó. Si es un duplicado, se descarta. Si llega un paquete con número de secuencia menor que el mayor visto hasta el momento, se rechaza como obsoleto debido que el enrutador tiene datos más recientes.

Este algoritmo tiene algunos problemas, pero son manejables. Primero, si los números de secuencia vuelven a comenzar, reinará la confusión. La solución aquí es utilizar un número de secuencia de 32 bits. Con un paquete de estado del enlace por segundo, el tiempo para volver a empezar será de 137 años, por lo que puede ignorarse esta posibilidad.

Segundo, si llega a caerse un enrutador, perderá el registro de su número de secuencia. Si comienza nuevamente en 0, se rechazará como duplicado el siguiente paquete.

Tercero, si llega a corromperse un número de secuencia y se escribe 65,540 en lugar de 4 (un error de 1 bit), los paquetes 5 a 65,540 serán rechazados como obsoletos, dado que se piensa que el número de secuencia actual es 65,540.

La solución a todos estos problemas es incluir la edad de cada paquete después del número de secuencia y disminuirla una vez cada segundo. Cuando la edad llega a cero, se descarta la información de ese enrutador. Generalmente, un paquete nuevo entra, por ejemplo, cada 10 segundos, por lo que la información de los enrutadores sólo expira cuando el enrutador está caído (o cuando se pierden seis paquetes consecutivos, evento poco probable). Los enrutadores también decrementan el campo de *edad* durante el proceso inicial de inundación para asegurar que no pueda perderse ningún paquete y sobrevivir durante un periodo indefinido (se descarta el paquete cuya edad sea cero).

Algunos refinamientos de este algoritmo lo hacen más robusto. Una vez que un paquete de estado del enlace llega a un enrutador para ser inundado, no se encola para transmisión inmediata. En vez de ello, entra en un área de almacenamiento donde espera un tiempo breve. Si antes de transmitirlo entra otro paquete de estado del enlace proveniente del mismo origen, se comparan sus números de secuencia. Si son iguales, se descarta el duplicado. Si son diferentes, se desecha el más viejo. Como protección contra los errores en las líneas enrutador-enrutador, se confirma la recepción de todos los paquetes de estado del enlace. Cuando se desactiva una línea, se examina el área de almacenamiento en orden *round-robin* para seleccionar un paquete o confirmación de recepción a enviar.

En la figura 5-14 se describe la estructura de datos usada por el enrutador *B* para la subred de la figura 5-13(a). Cada fila aquí corresponde a un paquete de estado del enlace recién llegado, pero aún no procesado por completo. La tabla registra dónde se originó el paquete, su número de secuencia y edad, así como los datos. Además, hay banderas de transmisión y de confirmación de recepción para cada una de las tres líneas de *B* (a *A*, *C* y *F*, respectivamente). Las banderas de envío significan que el paquete debe enviarse a través de la línea indicada. Las banderas de confirmación de recepción significan que su confirmación debe suceder ahí.

En la figura 5-14, el paquete de estado del enlace de *A* llegó directamente, por lo que debe enviarse a *C* y *F*, y debe confirmarse la recepción a *A*, como lo muestran los bits de bandera. De la misma manera, el paquete de *F* tiene que reenviarse a *A* y a *C*, y debe enviarse a *F* la confirmación de su recepción.

Sin embargo, la situación del tercer paquete, de *E*, es diferente. Llegó dos veces, la primera a través de *EAB* y la segunda por medio de *EFB*. En consecuencia, este paquete tiene que enviarse sólo a *C*, pero debe confirmarse su recepción tanto a *A* como a *F*, como lo indican los bits.

Si llega un duplicado mientras el original aún está en el búfer, los bits tienen que cambiar. Por ejemplo, si llega una copia del estado de *C* desde *F* antes de que se reenvíe la cuarta entrada de la tabla, cambiarán los seis bits a 100011 para indicar que debe enviarse a *F* una confirmación de recepción del paquete, sin enviarle el paquete mismo.

			Banderas de envío			Banderas de ACK			
Origen	Sec.	Edad	A	C	F	A	C	F	Datos
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Figura 5-14. El búfer de paquetes para el enrutador *B* de la figura 5-13.

Cálculo de las nuevas rutas

Una vez que un enrutador ha acumulado un grupo completo de paquetes de estado del enlace, puede construir el grafo de la subred completa porque todos los enlaces están representados. De hecho, cada enlace se representa dos veces, una para cada dirección. Los dos valores pueden promediarse o usarse por separado.

Ahora puede ejecutar localmente el algoritmo de Dijkstra para construir la ruta más corta a todos los destinos posibles. Los resultados de este algoritmo pueden instalarse en las tablas de enrutamiento, y la operación normal puede reiniciarse.

Para una subred con *n* enrutadores, cada uno de los cuales tiene *k* vecinos, la memoria requerida para almacenar los datos de entrada es proporcional a *kn*. En las subredes grandes éste puede ser un problema. También puede serlo el tiempo de cómputo. Sin embargo, en muchas situaciones prácticas, el enrutamiento por estado del enlace funciona bien.

Sin embargo, problemas con el hardware o el software pueden causar estragos con este algoritmo (lo mismo que con otros). Por ejemplo, si un enrutador afirma tener una línea que no tiene, u olvida una línea que sí tiene, el grafo de la subred será incorrecto. Si un enrutador deja de reenviar paquetes, o los corrompe al hacerlo, surgirán problemas. Por último, si al enrutador se le acaba la memoria o se ejecuta mal el algoritmo de cálculo de enrutamiento, surgirán problemas. A medida que la subred crece en decenas o cientos de miles de nodos, la probabilidad de falla ocasional de un enrutador deja de ser insignificante. Lo importante es tratar de limitar el daño cuando ocurra lo inevitable. Perlman (1988) estudia detalladamente estos problemas y sus soluciones.

El enrutamiento por estado del enlace se usa ampliamente en las redes actuales, por lo que son pertinentes unas pocas palabras sobre algunos protocolos que lo usan. El protocolo OSPF, que se emplea cada vez con mayor frecuencia en Internet, utiliza un algoritmo de estado del enlace. Describiremos OSPF en la sección 5.6.4.

Otro protocolo de estado del enlace importante es el **IS-IS** (**sistema intermedio-sistema intermedio**), diseñado por DECnet y más tarde adoptado por la ISO para utilizarlo con su protocolo de capa de red sin conexiones, CLNP. Desde entonces se ha modificado para manejar también

otros protocolos, siendo el más notable el IP. El IS-IS se usa en varias redes dorsales de Internet (entre ellas la vieja red dorsal NSFNET) y en muchos sistemas celulares digitales, como CDPD. El Novell NetWare usa una variante menor del IS-IS (NLSP) para el enrutamiento de paquetes IPX.

Básicamente, el IS-IS distribuye una imagen de la topología de enrutadores, a partir de la cual se calculan las rutas más cortas. Cada enrutador anuncia, en su información de estado del enlace, las direcciones de capa de red que puede alcanzar de manera directa. Estas direcciones pueden ser IP, IPX, AppleTalk o cualquier otra dirección. El IS-IS incluso puede manejar varios protocolos de red al mismo tiempo.

Muchas de las innovaciones diseñadas para el IS-IS fueron adoptadas por el OSPF (el cual se diseñó varios años después que el IS-IS). Entre éstas se encuentran un método autorregulable para inundar actualizaciones de estado del enlace, el concepto de un enrutador designado en una LAN y el método de cálculo y soporte de la división de rutas y múltiples métricas. En consecuencia, hay muy poca diferencia entre el IS-IS y el OSPF. La diferencia más importante es que el IS-IS está codificado de tal manera que es fácil y natural llevar de manera simultánea información sobre varios protocolos de capa de red, característica que no está presente en el OSPF. Esta ventaja es especialmente valiosa en entornos multiprotocolo grandes.

5.2.6 Enrutamiento jerárquico

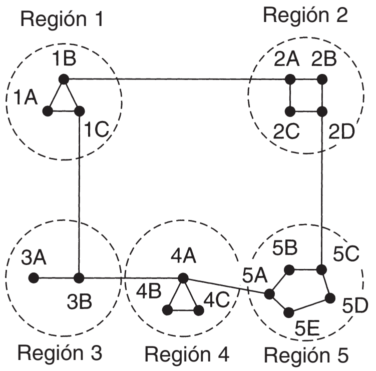
A medida que crece el tamaño de las redes, también lo hacen, de manera proporcional, las tablas de enrutamiento del enrutador. Las tablas que siempre crecen no sólo consumen memoria del enrutador, sino que también se necesita más tiempo de CPU para examinarlas y más ancho de banda para enviar informes de estado entre enrutadores. En cierto momento, la red puede crecer hasta el punto en que ya no es factible que cada enrutador tenga una entrada para cada uno de los demás enrutadores, por lo que el enrutamiento tendrá que hacerse de manera jerárquica, como ocurre en la red telefónica.

Cuando se utiliza el enrutamiento jerárquico, los enrutadores se dividen en lo que llamaremos **regiones,** donde cada enrutador conoce todos los detalles para enrutar paquetes a destinos dentro de su propia región, pero no sabe nada de la estructura interna de las otras regiones. Cuando se interconectan diferentes redes, es natural considerar cada una como región independiente a fin de liberar a los enrutadores de una red de la necesidad de conocer la estructura topológica de las demás.

En las redes enormes, una jerarquía de dos niveles puede ser insuficiente; tal vez sea necesario agrupar las regiones en clústeres, los clústeres en zonas, las zonas en grupos, etcétera, hasta que se nos agoten los nombres para clasificarlos. Como ejemplo de jerarquía multinivel, considere una posible forma de enrutar un paquete de Berkeley, California, a Malindi, Kenya. El enrutador de Berkeley conocería la topología detallada de California, pero podría enviar todo el tráfico exterior al enrutador de Los Ángeles. El enrutador de Los Ángeles podría enrutar el tráfico a otros enrutadores del país, pero enviaría el tráfico internacional a Nueva York. El enrutador de Nueva York tendría la programación para dirigir todo el tráfico al enrutador del país de destino encargado

del manejo de tráfico internacional, digamos en Nairobi. Por último, el paquete encontraría su camino por el árbol de Kenya hasta llegar a Malindi.

En la figura 5-15 se da un ejemplo cuantitativo de enrutamiento en una jerarquía de dos niveles con cinco regiones. La tabla de enrutamiento completa para el enrutador 1*A* tiene 17 entradas, como se muestra en la figura 5-15(b). Si el enrutamiento es jerárquico, como en la figura 5-15(c), hay entradas para todos los enrutadores locales, igual que antes, pero las demás regiones se han condensado en un solo enrutador, por lo que todo el tráfico para la región 2 va a través de la línea 1*B*-2*A*, pero el resto del tráfico remoto viaja por la línea 1*C*-3*B*. El enrutamiento jerárquico redujo la tabla de 17 entradas a 7. A medida que crece la razón entre la cantidad de regiones y el número de enrutadores por región, aumentan los ahorros de espacio de tabla.



(a)

Tabla completa para 1A

Dest.	Línea	Saltos
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(b)

Tabla jerárquica para 1A

Dest.	Línea	Saltos
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

Figura 5-15. Enrutamiento jerárquico.

Desgraciadamente, estas ganancias de espacio no son gratuitas. Se paga un precio, que es una longitud de ruta mayor. Por ejemplo, la mejor ruta de 1*A* a 5*C* es a través de la región 2 pero con el enrutamiento jerárquico, todo el tráfico a la región 5 pasa por la región 3, porque eso es mejor para la mayoría de los destinos de la región 5.

Cuando una red se vuelve muy grande, surge una pregunta interesante: ¿cuántos niveles debe tener la jerarquía? Por ejemplo, considere una subred con 720 enrutadores. Si no hay jerarquía, cada enrutador necesita 720 entradas en la tabla de enrutamiento. Si dividimos la subred en 24 regiones de 30 enrutadores cada una, cada enrutador necesitará 30 entradas locales más 23 entradas remotas, lo que da un total de 53 entradas. Si elegimos una jerarquía de tres niveles, con ocho

clústeres, cada uno de los cuales contiene 9 regiones de 10 enrutadores, cada enrutador necesita 10 entradas para los enrutadores locales, 8 entradas para el enrutamiento a otras regiones dentro de su propio clúster y 7 entradas para clústeres distantes, lo que da un total de 25 entradas. Kamoun y Kleinrock (1979) descubrieron que el número óptimo de niveles para una subred de N enrutadores es de ln N, y se requieren un total de e ln N entradas por enrutador. También han demostrado que el aumento en la longitud media efectiva de ruta causado por el enrutamiento jerárquico es tan pequeño que por lo general es aceptable.

5.2.7 Enrutamiento por difusión

En algunas aplicaciones, los *hosts* necesitan enviar mensajes a varios otros *hosts* o a todos los demás. Por ejemplo, el servicio de distribución de informes ambientales, la actualización de los precios de la bolsa o los programas de radio en vivo podrían funcionar mejor difundiéndolos a todas las máquinas y dejando que las que estén interesadas lean los datos. El envío simultáneo de un paquete a todos los destinos se llama **difusión;** se han propuesto varios métodos para llevarla a cabo.

Un método de difusión que no requiere características especiales de la subred es que el origen simplemente envíe un paquete distinto a todos los destinos. El método no sólo desperdicia ancho de banda, sino que también requiere que el origen tenga una lista completa de todos los destinos. En la práctica, ésta puede ser la única posibilidad, pero es el método menos deseable.

La inundación es otro candidato obvio. Aunque ésta es poco adecuada para la comunicación punto a punto ordinaria, para difusión puede merecer consideración seria, especialmente si no es aplicable ninguno de los métodos descritos a continuación. El problema de la inundación como técnica de difusión es el mismo que tiene como algoritmo de enrutamiento punto a punto: genera demasiados paquetes y consume demasiado ancho de banda.

Un tercer algoritmo es el **enrutamiento multidestino.** Con este método, cada paquete contiene una lista de destinos o un mapa de bits que indica los destinos deseados. Cuando un paquete llega al enrutador, éste revisa todos los destinos para determinar el grupo de líneas de salida que necesitará. (Se necesita una línea de salida si es la mejor ruta a cuando menos uno de los destinos.) El enrutador genera una copia nueva del paquete para cada línea de salida que se utilizará, e incluye en cada paquete sólo aquellos destinos que utilizarán la línea. En efecto, el grupo de destinos se divide entre las líneas de salida. Después de una cantidad suficiente de saltos, cada paquete llevará sólo un destino, así que puede tratarse como un paquete normal. El enrutamiento multidestino es como los paquetes con direccionamiento individual, excepto que, cuando varios paquetes deben seguir la misma ruta, uno de ellos paga la tarifa completa y los demás viajan gratis.

Un cuarto algoritmo de difusión usa explícitamente el árbol sumidero para el enrutador que inicia la difusión, o cualquier otro árbol de expansión adecuado. El **árbol de expansión** es un subgrupo de la subred que incluye todos los enrutadores pero no contiene ciclos. Si cada enrutador sabe cuáles de sus líneas pertenecen al árbol de expansión, puede copiar un paquete de entrada difundido en todas las líneas del árbol de expansión, excepto en aquella por la que llegó. Este método utiliza de manera óptima el ancho de banda, generando la cantidad mínima de paquetes necesarios para llevar a cabo el trabajo. El único problema es que cada enrutador debe tener cono-

cimiento de algún árbol de expansión para que este método pueda funcionar. Algunas veces esta información está disponible (por ejemplo, con el enrutamiento por estado del enlace), pero a veces no (por ejemplo, con el enrutamiento por vector de distancia).

Nuestro último algoritmo de difusión es un intento de aproximar el comportamiento del anterior, aun cuando los enrutadores no saben nada en lo absoluto sobre árboles de expansión. La idea, llamada **reenvío por ruta invertida** (*reverse path forwarding*), es excepcionalmente sencilla una vez planteada. Cuando llega un paquete difundido a un enrutador, éste lo revisa para ver si llegó por la línea normalmente usada para enviar paquetes *al* origen de la difusión. De ser así, hay excelentes posibilidades de que el paquete difundido haya seguido la mejor ruta desde el enrutador y, por lo tanto, sea la primera copia en llegar al enrutador. Si éste es el caso, el enrutador reenvía copias del paquete a todas las líneas, excepto a aquella por la que llegó. Sin embargo, si el paquete difundido llegó por una línea diferente de la preferida, el paquete se descarta como probable duplicado.

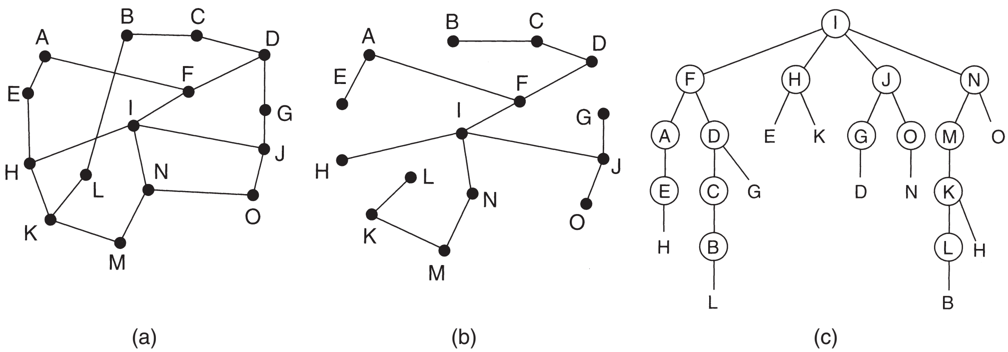


Figura 5-16. Reenvío por ruta invertida. (a) Subred. (b) Árbol sumidero. (c) Árbol construido mediante reenvío por ruta invertida.

En la figura 5-16 se muestra un ejemplo del reenvío por ruta invertida**.** En la parte (a) se muestra una subred, en la parte (b) se muestra un árbol sumidero para el enrutador *I* de esa subred, y en la parte (c) se muestra el funcionamiento del algoritmo de ruta invertida. En el primer salto, *I* envía paquetes a *F*, *H*, *J* y *N*, como lo indica la segunda fila del árbol. Cada uno de estos paquetes llega a *I* por la ruta preferida (suponiendo que la ruta preferida pasa a través del árbol sumidero), como lo indica el círculo alrededor de la letra. En el segundo salto, se generan ocho paquetes, dos por cada uno de los enrutadores que recibieron un paquete en el primer salto. Como resultado, los ocho llegan a enrutadores no visitados previamente, y cinco llegan a través de la línea preferida. De los seis paquetes generados en el tercer salto, sólo tres llegan por la ruta preferida (a *C, E* y *K*); los otros son duplicados. Después de cinco saltos y 24 paquetes, termina la difusión, en comparación con cuatro saltos y 14 paquetes si se hubiera seguido exactamente el árbol sumidero.

La ventaja principal del reenvío por ruta invertida es que es razonablemente eficiente y fácil de implementar. No requiere que los enrutadores conozcan los árboles de expansión ni tiene la sobrecarga de una lista de destinos o de un mapa de bits en cada paquete de difusión, como los

tiene el direccionamiento multidestino. Tampoco requiere mecanismos especiales para detener el proceso, como en el caso de la inundación (ya sea un contador de saltos en cada paquete y un conocimiento previo del diámetro de la subred, o una lista de paquetes ya vistos por origen).

5.2.8 Enrutamiento por multidifusión

Algunas aplicaciones requieren que procesos muy separados trabajen juntos en grupo; por ejemplo, un grupo de procesos que implementan un sistema de base de datos distribuido. En estos casos, con frecuencia es necesario que un proceso envíe un mensaje a todos los demás miembros del grupo. Si el grupo es pequeño, simplemente se puede transmitir a cada uno de los miembros un mensaje punto a punto. Si el grupo es grande, esta estrategia es costosa. A veces puede usarse la difusión, pero su uso para informar a 1000 máquinas de una red que abarca un millón de nodos es ineficiente porque la mayoría de los receptores no están interesados en el mensaje (o, peor aún, definitivamente están interesados pero no deben verlo). Por lo tanto, necesitamos una manera de enviar mensajes a grupos bien definidos de tamaño numéricamente grande, pero pequeños en comparación con la totalidad de la red.

El envío de un mensaje a uno de tales grupos se llama **multidifusión,** y su algoritmo de enrutamiento es el **enrutamiento por multidifusión.** En esta sección describiremos una manera de lograr el enrutamiento por multidifusión. Para información adicional, vea (Chu y cols., 2000; Costa y cols., 2001; Kasera y cols., 2000; Madruga y García-Luna-Aceves, 2001, y Zhang y Ryu, 2001).

Para la multidifusión se requiere administración de grupo. Se necesita alguna manera de crear y destruir grupos, y un mecanismo para que los procesos se unan a los grupos y salgan de ellos. La forma de realizar estas tareas no le concierne al algoritmo de enrutamiento. Lo que sí le concierne es que cuando un proceso se una a un grupo, informe a su *host* este hecho. Es importante que los enrutadores sepan cuáles de sus *hosts* pertenecen a qué grupos. Los *hosts* deben informar a sus enrutadores de los cambios en los miembros del grupo, o los enrutadores deben enviar de manera periódica la lista de sus *hosts*. De cualquier manera, los enrutadores aprenden qué *hosts* pertenecen a cuáles grupos. Los enrutadores les dicen a sus vecinos, de manera que la información se propaga a través de la subred.

Para realizar enrutamiento de multidifusión, cada enrutador calcula un árbol de expansión que cubre a todos los demás enrutadores de la subred. Por ejemplo, en la figura 5-17(a) tenemos una subred con dos grupos, 1 y 2. Algunos enrutadores están conectados a *hosts* que pertenecen a uno o ambos grupos, como se indica en la figura. En la figura 5-17(b) se muestra un árbol de expansión para el enrutador de la izquierda.

Cuando un proceso envía un paquete de multidifusión a un grupo, el primer enrutador examina su árbol de expansión y lo recorta, eliminando todas las líneas que conduzcan a *hosts* que no sean miembros del grupo. En el ejemplo de la figura 5-17(c) se muestra el árbol de expansión recortado del grupo 1. De la misma manera, en la figura 5-17(d) se presenta el árbol de expansión recortado del grupo 2. Los paquetes de multidifusión se reenvían sólo a través del árbol de expansión apropiado.

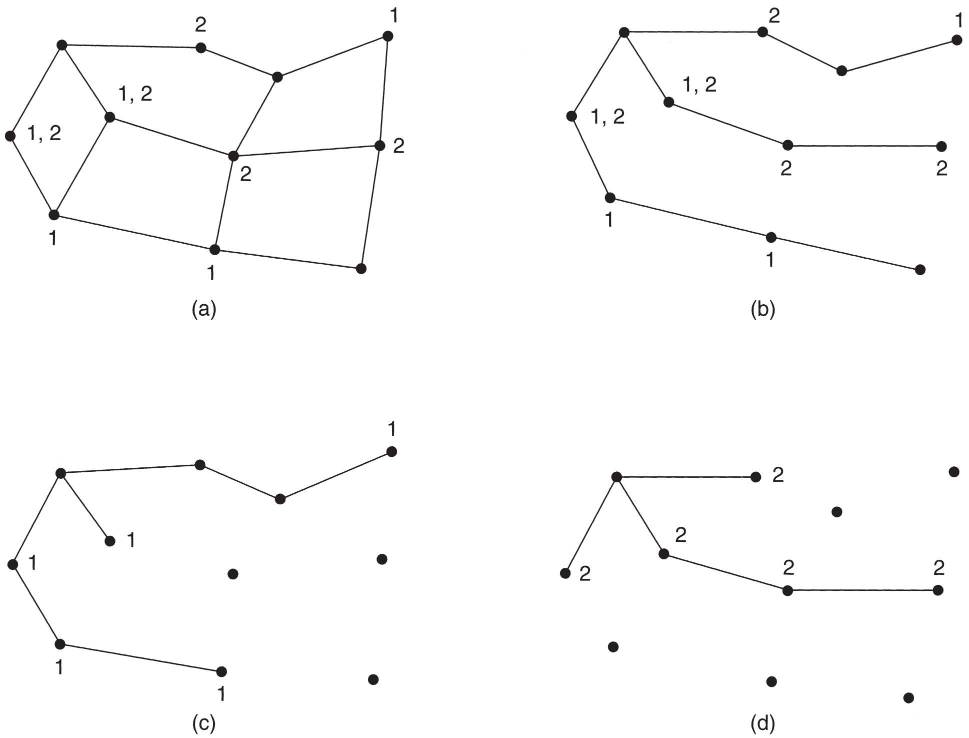


Figura 5-17. (a) Subred. (b) Árbol de expansión del enrutador del extremo izquierdo. (c) Árbol de multidifusión del grupo 1. (d) Árbol de multidifusión para el grupo 2.

Hay varias maneras de recortar el árbol de expansión. La más sencilla puede utilizarse si se maneja enrutamiento por estado del enlace, y cada enrutador está consciente de la topología completa de la subred, incluyendo qué *hosts* pertenecen a cuáles grupos. Después puede recortarse el árbol, comenzando por el final de cada ruta y trabajando hacia la raíz, eliminando todos los enrutadores que no pertenecen al grupo en cuestión.

Con el enrutamiento por vector de distancia se puede seguir una estrategia de recorte diferente. El algoritmo básico es el reenvío por ruta invertida. Sin embargo, cuando un enrutador sin *hosts* interesados en un grupo en particular y sin conexiones con otros enrutadores recibe un mensaje de multidifusión para ese grupo, responde con un mensaje de recorte (PRUNE), indicando al emisor que no envíe más multidifusiones para ese grupo. Cuando un enrutador que no tiene miembros del grupo entre sus propios *hosts* recibe uno de tales mensajes por todas las líneas, también puede responder con un mensaje de recorte. De esta forma, la subred se recorta recursivamente.

Una desventaja potencial de este algoritmo es que no escala bien en redes grandes. Suponga que una red tiene *n* grupos, cada uno con un promedio de *m* miembros. Por cada grupo se deben almacenar *m* árboles de expansión recortados, lo que da un total de *mn* árboles. Cuando hay muchos grupos grandes, se necesita bastante espacio para almacenar todos estos árboles.

Un diseño alterno utiliza **árboles de núcleo** (*core-based trees*) (Ballardie y cols., 1993). Aquí se calcula un solo árbol de expansión por grupo, con la raíz (el núcleo) cerca de la mitad del grupo. Para enviar un mensaje de multidifusión, un *host* lo envía al núcleo, que entonces hace la multidifusión a través del árbol de expansión. Aunque este árbol no será óptimo para todos los orígenes, la reducción de costos de almacenamiento de *m* árboles a un árbol por grupo representa un ahorro significativo.

5.2.9 Enrutamiento para *hosts* móviles

En la actualidad millones de personas tienen computadoras portátiles, y generalmente quieren leer su correo electrónico y acceder a sus sistemas de archivos normales desde cualquier lugar del mundo. Estos *hosts* móviles generan una nueva complicación: para enrutar un paquete a un *host* móvil, la red primero tiene que encontrarlo. El tema de la incorporación de *hosts* móviles en una red es muy reciente, pero en esta sección plantearemos algunos de los problemas relacionados y sugeriremos una posible solución.

En la figura 5-18 se muestra el modelo del mundo que usan generalmente los diseñadores de red. Aquí tenemos una WAN que consiste en enrutadores y *hosts*. Conectadas a la WAN hay varias LANs, MANs y celdas inalámbricas del tipo que estudiamos en el capítulo 2.

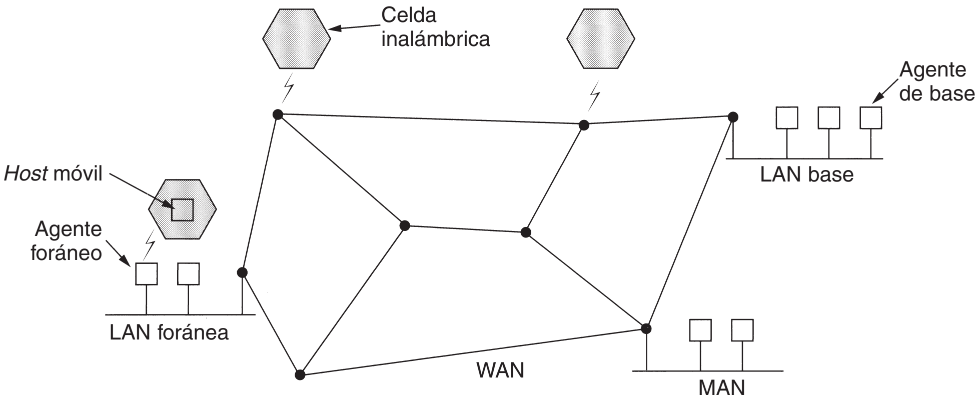


Figura 5-18. WAN a la que están conectadas LANs, MANs y celdas inalámbricas.

Se dice que los *hosts* que nunca se mueven son estacionarios; se conectan a la red mediante cables de cobre o fibra óptica. En contraste, podemos distinguir otros dos tipos de *hosts*. Los *hosts* migratorios básicamente son *hosts* estacionarios que se mueven de un lugar fijo a otro de tiempo en tiempo, pero que usan la red sólo cuando están conectados físicamente a ella. Los *hosts* ambulantes hacen su cómputo en movimiento, y necesitan mantener sus conexiones mientras se trasladan de un lado a otro. Usaremos el término ***hosts* móviles** para referirnos a cualquiera de las dos últimas categorías, es decir, a todos los *hosts* que están lejos de casa y que necesitan seguir conectados.

Se supone que todos los *hosts* tienen una **localidad base** que nunca cambia. Los *hosts* también tienen una dirección base permanente que puede servir para determinar su localidad base, de manera análoga a como el número telefónico 1-212-5551212 indica Estados Unidos (código de país 1) y Manhattan (212). La meta de enrutamiento en los sistemas con *hosts* móviles es posibilitar el envío de paquetes a *hosts* móviles usando su dirección base, y hacer que los paquetes lleguen eficientemente a ellos en cualquier lugar en el que puedan estar. Lo difícil, por supuesto, es encontrarlos.

En el modelo de la figura 5-18, el mundo se divide (geográficamente) en unidades pequeñas, a las que llamaremos áreas. Un área por lo general es una LAN o una celda inalámbrica. Cada área tiene uno o más **agentes foráneos,** los cuales son procesos que llevan el registro de todos los *hosts* móviles que visitan el área. Además, cada área tiene un **agente de base,** que lleva el registro de todos los *hosts* cuya base está en el área, pero que actualmente están visitando otra área.

Cuando un nuevo *host* entra en un área, ya sea al conectarse a ella (por ejemplo, conectándose a la LAN), o simplemente al entrar en la celda, su computadora debe registrarse con el agente foráneo de ese lugar. El procedimiento de registro funciona típicamente de esta manera:

1. Periódicamente, cada agente foráneo difunde un paquete que anuncia su existencia y dirección. Un *host* móvil recién llegado puede esperar uno de estos mensajes, pero si no llega ninguno con suficiente rapidez, el *host* móvil puede difundir un paquete que diga: “¿hay agentes foráneos por ahí?”
2. El *host* móvil se registra con el agente foráneo, dando su dirección base, su dirección actual de capa de enlace de datos y cierta información de seguridad.
3. El agente foráneo se pone en contacto con el agente de base del *host* móvil y le dice: “uno de tus *hosts* está por aquí”. El mensaje del agente foráneo al agente de base contiene la dirección de red del agente foráneo, así como la información de seguridad, para convencer al agente de base de que el *host* móvil en realidad está ahí.
4. El agente de base examina la información de seguridad, que contiene una marca de tiempo, para comprobar que fue generada en los últimos segundos. Si está conforme, indica al agente foráneo que proceda.
5. Cuando el agente foráneo recibe la confirmación de recepción del agente de base, hace una entrada en sus tablas e informa al *host* móvil que ahora está registrado.

Idealmente, cuando un *host* sale de un área, este hecho también se debe anunciar para permitir que se borre el registro, pero muchos usuarios apagan abruptamente sus computadoras cuando terminan.

Cuando un paquete se envía a un *host* móvil, se enruta a la LAN base del *host*, ya que eso es lo indicado en la dirección, como se ilustra en el paso 1 de la figura 5-19. El emisor, que se encuentra en la ciudad noreste de Seattle, desea enviar un paquete a un *host* que se encuentra normalmente en Nueva York. Los paquetes enviados al *host* móvil en su LAN base de Nueva York son interceptados por el agente de base que se encuentra ahí. A continuación, dicho agente busca la

nueva ubicación (temporal) del *host* móvil y encuentra la dirección del agente foráneo que maneja al *host* móvil, en Los Ángeles.

El agente de base entonces hace dos cosas. Primero, encapsula el paquete en el campo de carga útil de un paquete exterior y envía este último al agente foráneo (paso 2 de la figura 5-19). Este mecanismo se llama entunelamiento y lo veremos con mayor detalle después. Tras obtener el paquete encapsulado, el agente foráneo extrae el paquete original del campo de carga útil y lo envía al *host* móvil como trama de enlace de datos.

Segundo, el agente de base indica al emisor que en lo futuro envíe paquetes al *host* móvil encapsulándolos en la carga útil de paquetes explícitamente dirigidos al agente foráneo, en lugar de simplemente enviarlos a la dirección base del *host* móvil (paso 3). Los paquetes subsiguientes ahora pueden enrutarse en forma directa al usuario por medio del agente foráneo (paso 4), omitiendo la localidad base por completo.

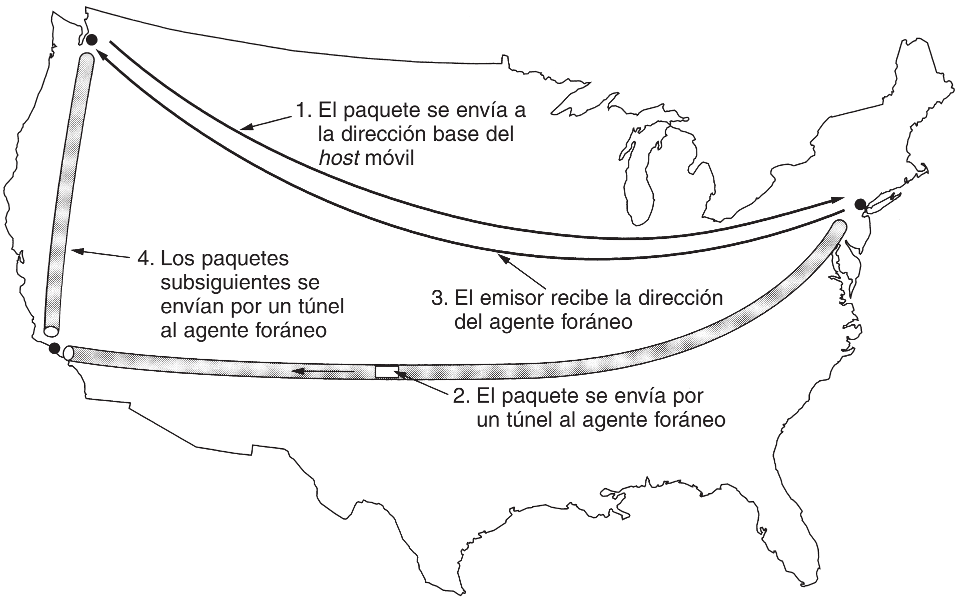


Figura 5-19. Enrutamiento de paquetes para *hosts* móviles.

Los diferentes esquemas propuestos difieren en varios sentidos. Primero está el asunto de qué parte del protocolo es llevada a cabo por los enrutadores y cuál por los *hosts* y, en este último caso, por cuál capa de los *hosts*. Segundo, en unos cuantos esquemas, los enrutadores a lo largo del camino registran las direcciones asignadas, para poder interceptarlas y redirigir el tráfico aún antes de que llegue a la dirección base. Tercero, en algunos esquemas, cada visitante recibe una dirección temporal única; en otros, la dirección temporal se refiere a un agente que maneja el tráfico de todos los visitantes.

Cuarto, los esquemas difieren en la manera en que logran realmente que los paquetes dirigidos a un destino lleguen a uno diferente. Una posibilidad es cambiar la dirección de destino y simplemente retransmitir el paquete modificado. Como alternativa, el paquete completo, con dirección base y todo, puede encapsularse en la carga útil de otro paquete enviado a la dirección temporal. Por último, los esquemas difieren en sus aspectos de seguridad. Por lo general, cuando un *host* o un enrutador recibe un mensaje del tipo "a partir de este momento, envíame por favor todo el correo de Genoveva", puede tener dudas acerca de con quién está hablando y de si se trata de una buena idea. En (Hac y Guo, 2000; Perkins, 1998a; Snoeren y Balakrishnan, 2000; Solomon, 1998, y Wang y Chen, 2001), se analizan y comparan varios protocolos para *hosts* móviles.

5.2.10 Enrutamiento en redes *ad hoc*

Ya vimos cómo realizar el enrutamiento cuando los *hosts* son móviles y los enrutadores son fijos. Un caso aún más extremo es uno en el que los enrutadores mismos son móviles. Entre las posibilidades se encuentran:

1. Vehículos militares en un campo de batalla sin infraestructura.
2. Una flota de barcos en el mar.
3. Trabajadores de emergencia en un área donde un temblor destruyó la infraestructura.
4. Una reunión de personas con computadoras portátiles en un área que no cuenta con 802.11.

En todos estos casos, y en otros, cada nodo consiste en un enrutador y un *host*, por lo general en la misma computadora. Las redes de nodos que están cerca entre sí se conocen como **redes *ad hoc*** o **MANETs** (**Redes *ad hoc* Móviles**). A continuación las examinaremos con brevedad. Para mayor información, vea (Perkins, 2001).

Lo que distingue a las redes *ad hoc* de las redes cableadas es que en las primeras se eliminaron todas las reglas comunes acerca de las topologías fijas, los vecinos fijos y conocidos, la relación fija entre direcciones IP y la ubicación, etcétera. Los enrutadores pueden ir y venir o aparecer en nuevos lugares en cualquier momento. En una red cableada, si un enrutador tiene una ruta válida a algún destino, esa ruta continúa siendo válida de manera indefinida (excepto si ocurre una falla en alguna parte del sistema que afecte a esa ruta). En una red *ad hoc*, la topología podría cambiar todo el tiempo, por lo que la necesidad o la validez de las rutas puede cambiar en cualquier momento, sin previo aviso. No es necesario decir que estas circunstancias hacen del enrutamiento en redes *ad hoc* algo diferente del enrutamiento en sus contrapartes fijas.

Se ha propuesto una variedad de algoritmos de enrutamiento para las redes *ad hoc*. Uno de los más interesantes es el algoritmo de enrutamiento **AODV** (**Vector de Distancia *ad hoc* bajo Demanda**) (Perkins y Royer, 1999). Es pariente lejano del algoritmo de vector de distancia Bellman-Ford pero está adaptado para funcionar en entornos móviles y toma en cuenta el ancho de banda

limitado y la duración corta de la batería en esos entornos. Otra característica inusual es que es un algoritmo bajo demanda, es decir, determina una ruta a algún destino sólo cuando alguien desea enviar un paquete a ese destino. A continuación veremos lo que eso significa.

Descubrimiento de ruta

En cualquier instante dado, una red *ad hoc* puede describirse mediante un grafo de los nodos (enrutadores + *hosts*). Dos nodos se conectan (es decir, tienen un arco entre ellos en el grafo) si se pueden comunicar de manera directa mediante sus radios. Debido a que uno de los dos podría tener un emisor más poderoso que el otro, es posible que *A* esté conectado a *B*, pero *B* no está conectado a *A*. Sin embargo, por simplicidad, asumiremos que todas las conexiones son simétricas. También debe hacerse notar que el simple hecho de que dos nodos estén dentro del alcance de radio entre sí no significa que estén conectados. Puede haber edificios, colinas u otros obstáculos que bloqueen su comunicación.

Para describir el algoritmo, considere la red *ad hoc* de la figura 5-20, en la que un proceso del nodo *A* desea enviar un paquete al nodo *I*. El algoritmo AODV mantiene una tabla en cada nodo, codificada por destino, que proporciona información acerca de ese destino, incluyendo a cuál vecino enviar los paquetes a fin de llegar al destino. Suponga que *A* busca en sus tablas y no encuentra una entrada para *I*. Ahora tiene que descubrir una ruta a *I*. Debido a esta propiedad de descubrir rutas sólo cuando es necesario este algoritmo se conoce como “bajo demanda”.

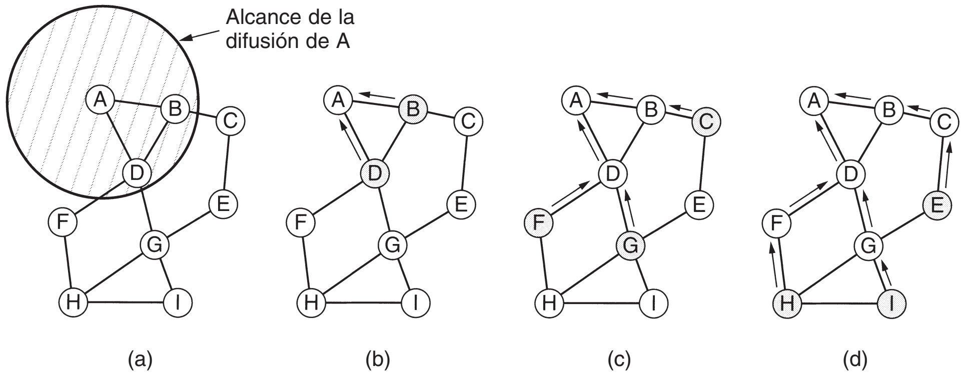


Figura 5-20. (a) Alcance de la difusión de *A*. (b) Después de que *B* y *D* reciben la difusión de *A*. (c) Después de que *C*, *F* y *G* reciben la difusión de *A*. (d) Después de que *E*, *H* e *I* reciben la difusión de *A*. Los nodos sombreados son nuevos receptores. Las flechas muestran las rutas invertidas posibles.

Para localizar a *I*, *A* construye un paquete especial de solicitud de ruta (ROUTE REQUEST) y lo difunde. El paquete llega a *B* y *D*, como se ilustra en la figura 5-20(a). De hecho, la razón por la que *B* y *D* se conectan a *A* en el grafo es que pueden recibir comunicación de *A*. Por ejemplo,

F no se muestra con un arco a *A* porque no puede recibir la señal de radio de *A*. Por lo tanto, *F* no se conecta a *A*.

En la figura 5-21 se muestra el formato del paquete de solicitud de ruta. Contiene las direcciones de origen y destino, por lo general sus direcciones IP, mediante las cuales se identifica quién está buscando a quién. También contiene un *ID de solicitud*, que es un contador local que se mantiene por separado en cada nodo y se incrementa cada vez que se difunde un paquete de solicitud de ruta. En conjunto, los campos *Dirección de origen* e *ID de solicitud* identifican de manera única el paquete de solicitud de ruta a fin de que los nodos descarten cualquier duplicado que pudieran recibir.

Dirección de origen	ID de solicitud	Dirección de destino	# de secuencia de origen	# de secuencia de destino	Cuenta de saltos

Figura 5-21. Formato de un paquete ROUTE REQUEST.

Además del contador *ID de solicitud*, cada nodo también mantiene un segundo contador de secuencia que se incrementa cada vez que se envía un paquete de solicitud de ruta (o una respuesta al paquete de solicitud de ruta de alguien más). Funciona de forma muy parecida a un reloj y se utiliza para indicar nuevas rutas a partir de rutas anteriores. El cuarto campo de la figura 5-21 es un contador de secuencia de *A*; el quinto campo es el valor más reciente del número de secuencia de *I* que *A* ha visto (0 si nunca lo ha visto). El uso de estos campos se aclarará pronto. El último campo, *Cuenta de saltos*, mantendrá un registro de cuántos saltos ha realizado el paquete. Se inicializa a 0.

Cuando un paquete de solicitud de ruta llega a un nodo (*B* y *D* en este caso), se procesa mediante los siguientes pasos.

1. El par (*Dirección de origen*, *ID de solicitud*) se busca en una tabla de historia local para ver si esta solicitud ya se había visto y procesado. Si es un duplicado, se descarta y el procesamiento se detiene. Si no es un duplicado, el par se introduce en la tabla de historia a fin de que se puedan rechazar futuros duplicados, y el procesamiento continúa.

2. El receptor busca el destino en su tabla de enrutamiento. Si se conoce una ruta reciente al destino, se regresa un paquete de respuesta de ruta (ROUTE REPLY) al origen que le indica cómo llegar al destino (básicamente: utilízame). Reciente significa que el *Número de secuencia de destino* almacenado en la tabla de enrutamiento es mayor que o igual al *Número de secuencia de destino* del paquete de solicitud de ruta. Si es menor, la ruta almacenada es más antigua que la que el origen tenía para el destino, por lo que se ejecuta el paso 3.

3. Puesto que el receptor no conoce una ruta reciente al destino, incrementa el campo *Cuenta de saltos* y vuelve a difundir el paquete de solicitud de ruta. También extrae los datos del paquete y los almacena como una entrada nueva en su tabla de rutas invertidas. Esta información se utilizará para construir la ruta invertida a fin de que la respuesta pueda

regresar posteriormente al origen. Las flechas que se muestran en la figura 5-20 se utilizan para construir la ruta invertida. También se inicia un temporizador para la nueva entrada de ruta invertida. Si expira, la entrada se borra.

Ni *B* ni *D* saben en dónde está *I*, por lo tanto, cada uno de estos nodos crea una entrada de ruta invertida que apunta a *A*, como lo muestran las flechas de la figura 5-20, y difunde el paquete con la *Cuenta de saltos* establecida a 1. La difusión de *B* llega a *C* y *D*. *C* crea una entrada para él en su tabla de rutas invertidas y la vuelve a difundir. En contraste, *D* la rechaza como un duplicado. De manera similar, *B* rechaza la difusión de *D*. Sin embargo, *F* y *G* aceptan la difusión de *D* y la almacenan como se muestra en la figura 5-20(c). Después de que *E*, *H* e *I* reciben la difusión, el paquete de solicitud de ruta finalmente alcanza un destino que sabe dónde está *I*, en este caso *I* mismo, como se ilustra en la figura 5-20(d). Observe que aunque mostramos las difusiones en tres pasos separados, las difusiones de nodos diferentes no se coordinan de ninguna forma.

En respuesta a la solicitud entrante, *I* construye un paquete de respuesta de ruta, como se muestra en la figura 5-22. La *Dirección de origen*, *Dirección de destino* y *Cuenta de saltos* se copian de la solicitud entrante, pero el *Número de secuencia de destino* se toma de su contador en memoria. El campo *Cuenta de saltos* se establece a 0. El campo *Tiempo de vida* controla cuánto tiempo es válida la ruta. Este paquete se difunde únicamente al nodo de donde proviene la solicitud de ruta, que en este caso es *G*. Después sigue la ruta invertida a *D* y, finalmente, a *A*. En cada nodo se incrementa *Cuenta de saltos* de modo que el nodo puede ver a qué distancia está del destino (*I*).

Dirección de origen	Dirección de destino	# de secuencia de origen	Cuenta de saltos	Tiempo de vida

Figura 5-22. Formato de un paquete de respuesta de ruta.

El paquete se inspecciona en cada nodo intermedio del camino de regreso. Se introduce en la tabla de enrutamiento local como una ruta a *I* si se cumple una o más de las siguientes tres condiciones:

1. No se conoce una ruta a *I*.
2. El número de secuencia para *I* en el paquete de respuesta de ruta es mayor que el valor en la tabla de enrutamiento.
3. Los números de secuencia son iguales pero la nueva ruta es más corta.

De esta forma, todos los nodos de la ruta invertida aprenden gratis la ruta a *I*, gracias al descubrimiento de ruta de *A*. Los nodos que obtuvieron el paquete original de solicitud de ruta pero que no estaban en la ruta invertida (*B*, *C*, *E*, *F* y *H* en este ejemplo) descartan la entrada de la tabla de ruta invertida cuando el temporizador asociado termina.

En una red grande, el algoritmo genera muchas difusiones, incluso para los destinos que están cerca. El número de difusiones se puede reducir como se muestra a continuación. El campo *Tiempo de vida* del paquete IP se inicializa al diámetro esperado de la red y se decrementa en cada salto. Si llega a 0, el paquete se descarta en lugar de difundirse.

El proceso de descubrimiento se modifica como se muestra a continuación. Para localizar un destino, el emisor difunde un paquete de solicitud de ruta con el campo *Tiempo de vida* establecido a 1. Si dentro de un tiempo razonable no se obtiene respuesta alguna, se envía otro paquete, pero esta vez con el campo *Tiempo de vida* establecido a 2. Los intentos subsiguientes utilizan 3, 4, 5, etcétera. De esta forma, la búsqueda primero se intenta de manera local y, después, en anillos cada vez más grandes.

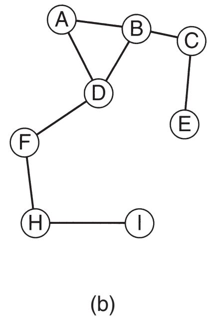
Mantenimiento de rutas

Debido a que es posible mover o apagar los nodos, la topología puede cambiar de manera espontánea. Por ejemplo, en la figura 5-20, si *G* se apaga, *A* no se dará cuenta de que la ruta a *I* (*ADGI*) que estaba utilizando ya no es válida. El algoritmo necesita ser capaz de manejar esto. Cada nodo difunde de manera periódica un mensaje de saludo (*Hello*). Se espera que cada uno de sus vecinos responda a dicho mensaje. Si no se recibe ninguna respuesta, el difusor sabe que el vecino se ha movido del alcance y ya no está conectado a él. De manera similar, si el difusor trata de enviar un paquete a un vecino que no responde, se da cuenta de que el vecino ya no está disponible.

Esta información se utiliza para eliminar rutas que ya no funcionan. Para cada destino posible, cada nodo, *N*, mantiene un registro de sus vecinos que le han proporcionado un paquete para ese destino durante los últimos ΔT segundos. Éstos se llaman **vecinos activos** de *N* para ese destino. *N* hace esto mediante una tabla de enrutamiento codificada por destino y al contener el nodo de salida a utilizar para llegar al destino, la cuenta de saltos al destino, el número de secuencia de destino más reciente y la lista de vecinos activos para ese destino. En la figura 5-23(a) se muestra una tabla de enrutamiento posible para el nodo *D* en nuestra topología de ejemplo.

Dest.	Siguiente salto	Distancia	Vecinos activos	Otros campos
A	A	1	F, G	
B	B	1	F, G	
C	B	2	F	
E	G	2		
F	F	1	A, B	
G	G	1	A, B	
H	F	2	A, B	
I	G	2	A, B	

(a)



(b)

Figura 5-23. (a) La tabla de enrutamiento de *D* antes de que *G* se apague. (b) El grafo después de que *G* se ha apagado.

Cuando cualquiera de los vecinos de *N* se vuelve inalcanzable, *N* verifica su tabla de enrutamiento para ver cuáles destinos tienen rutas en las que se incluya al vecino ahora perdido. Para cada una de estas rutas, se les informa a los vecinos activos que su ruta a través de *N* ahora es inválida y que se debe eliminar de sus tablas de enrutamiento. A continuación los vecinos activos indican este hecho a sus vecinos activos, y así sucesivamente y de manera recursiva, hasta que las rutas que dependen del nodo perdido se eliminan de todas las tablas de enrutamiento.

Como ejemplo del mantenimiento de ruta, considere nuestro ejemplo previo, pero ahora *G* se apaga de manera repentina. En la figura 5-23(b) se ilustra la topología modificada. Cuando *D* descubre que *G* ha desaparecido, busca en su tabla de enrutamiento y ve que *G* se utilizó en rutas a *E*, *G* e *I*. La unión de los vecinos activos para estos destinos es el conjunto {*A*, *B*}. En otras palabras, *A* y *B* dependen de *G* para algunas de sus rutas, y por esto se les tiene que informar que estas rutas ya no funcionan. *D* les indica este hecho enviándoles paquetes que causan que actualicen sus propias tablas de enrutamiento de manera acorde. *D* también elimina las entradas de *E*, *G* e *I* de su tabla de enrutamiento.

En nuestra descripción tal vez no sea obvio, pero una diferencia importante entre AODV y Bellman-Ford es que los nodos no envían difusiones periódicas que contengan su tabla de enrutamiento completa. Esta diferencia ahorra ancho de banda y duración de batería.

AODV también es capaz de realizar enrutamiento de difusión y multidifusión. Para mayores detalles, consulte (Perkins y Royer, 2001). El enrutamiento *ad hoc* es un área de investigación reciente. Se ha escrito bastante sobre este tema. Algunas de las obras son (Chen y cols., 2002; Hu y Johnson, 2001; Li y cols., 2001; Raju y Garcia-Luna-Aceves, 2001; Ramanathan y Redi, 2002; Royer y Toh, 1999; Spohn y Garcia-Luna-Aceves, 2001; Tseng y cols., 2001, y Zadeh y cols., 2002).

5.2.11 Búsqueda de nodos en redes de igual a igual

Las redes de igual a igual son un fenómeno relativamente nuevo, en las cuales una gran cantidad de personas, por lo general con conexiones cableadas permanentes a Internet, están en contacto para compartir recursos. La primera aplicación de uso amplio de la tecnología de igual a igual sirvió para cometer un delito masivo: 50 millones de usuarios de Napster intercambiaron canciones con derechos de autor sin el permiso de los poseedores de tales derechos hasta que las cortes cerraron Napster en medio de una gran controversia. Sin embargo, la tecnología de igual a igual tiene muchos usos interesantes y legales. También tiene algo similar a un problema de enrutamiento, aunque no como los que hemos estudiado hasta el momento. No obstante, vale la pena echarle un vistazo breve.

Lo que hace que los sistemas de igual a igual sean interesantes es que son totalmente distribuidos. Todos los nodos son simétricos y no hay control central o jerarquía. En un sistema típico de igual a igual, cada usuario tiene algo de información que podría ser de interés para otros usuarios. Esta información podría ser software (del dominio público), música, fotografías, etcétera, todos gratuitos. Si hay una gran cantidad de usuarios, éstos no se conocerán entre sí y no sabrán en dónde buscar lo que desean. Una solución es una base de datos central enorme, pero tal vez esto no sea tan factible por alguna razón (por ejemplo, nadie está dispuesto a albergarla ni a mantenerla). Por

lo tanto, el problema se reduce a qué debe hacer el usuario para encontrar un nodo que contiene lo que desea cuando no hay una base de datos centralizada o incluso un índice centralizado.

Supongamos que cada usuario tiene uno o más elementos de datos, como canciones, fotografías, programas, archivos, etcétera, que otros usuarios podrían querer leer. Cada elemento tiene una cadena ASCII que lo nombra. Un usuario potencial sólo conoce la cadena ASCII y desea averiguar si una o más personas tienen copias, y de ser así, cuáles son sus direcciones IP.

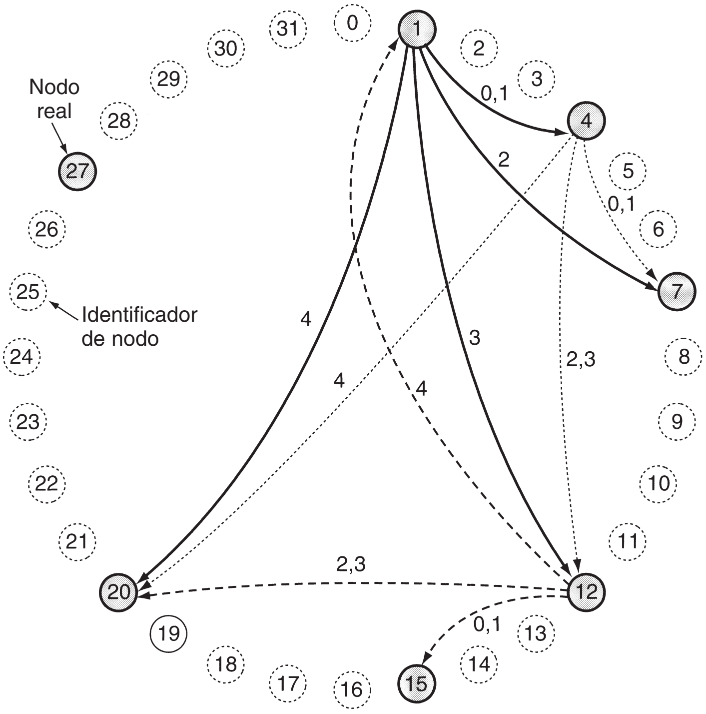
Como ejemplo, considere una base de datos genealógica distribuida. Cada genealogista tiene algunos registros en línea de sus predecesores y parientes, posiblemente con fotos, audio o incluso videoclips de las personas. Varias personas pueden tener el mismo bisabuelo, por lo que un predecesor podría tener registros en múltiples nodos. El nombre del registro es el nombre de la persona de alguna forma canónica. En algún punto, un genealogista descubre el testamento de su bisabuelo en un archivo, en el que su bisabuelo hereda su reloj de bolsillo de oro a su sobrino. El genealogista ahora sabe el nombre del sobrino y desea averiguar si otros genealogistas tienen un registro de dicho sobrino. ¿De qué manera sería posible, sin una base de datos central, saber quién, en caso de que lo hubiera, lo tiene?

Se han propuesto varios algoritmos para resolver este problema. El que examinaremos se llama Chord (Dabek y cols., 2001a, y Stoica y cols., 2001). A continuación se proporciona una explicación simplificada de cómo funciona. El sistema Chord consiste de *n* usuarios participantes, cada uno de los cuales podría contar con algunos registros almacenados y además está preparado para almacenar bits y fragmentos del índice para que otros usuarios los utilicen. Cada nodo de usuario tiene una dirección IP que puede generar un código de *hash* de *m* bits mediante una función de *hash*. Chord utiliza SHA-1 para *hash*. SHA-1 se utiliza en la criptografía; en el capítulo 8 analizaremos este tema. Por ahora, sólo es una función que toma como argumento una cadena de bytes de longitud variable y produce un número completamente aleatorio de 160 bits. Por lo tanto, podemos convertir cualquier dirección IP a un número de 160 bits llamado **identificador de nodo.**

Conceptualmente, todos los 2^{160} identificadores de nodos están organizados en orden ascendente en un gran círculo. Algunos de ellos corresponden a nodos participantes, pero la mayoría no. En la figura 5-24(a) mostramos el círculo de identificador de nodo de $m = 5$ (por el momento ignore el arco de en medio). En este ejemplo, los nodos con identificadores 1, 4, 7, 12, 15, 20 y 27 corresponden a nodos reales y están sombreados en la figura; el resto no existe.

A continuación definiremos la función *sucesor*(*k*) como el identificador de nodo del primer nodo real que sigue a *k* alrededor del círculo en el sentido de las manecillas del reloj. Por ejemplo, *sucesor*(6) = 7, *sucesor*(8) = 12 y *sucesor*(22) = 27.

A los nombres de los registros (nombres de canciones, nombres de los predecesores, etcétera) también se les aplica la función de *hash* (es decir, SHA-1) para generar un número de 160 bits, llamado **clave.** Por lo tanto, para convertir *nombre* (el nombre ASCII del registro) a su clave, utilizamos *clave* = *hash*(*nombre*). Este cálculo es simplemente una llamada de procedimiento local a *hash.* Si una persona que posee un registro genealógico para *nombre* desea ponerlo a disposición de todos, primero construye una tupla que consiste de (*nombre*, *mi-dirección-IP*) y después solicita a *sucesor*(*hash*(*nombre*)) que almacene la tupla. Si existen múltiples registros (en nodos diferentes) para este nombre, su tupla se almacenará en el mismo nodo. De esta forma, el índice se



(a)

Inicio	Dirección IP del sucesor
2	4
3	4
5	7
9	12
17	20

Tabla finger del nodo 1

Inicio	Dirección IP del sucesor
5	7
6	7
8	12
12	12
20	20

Tabla finger del nodo 4

Inicio	Dirección IP del sucesor
13	15
14	15
16	20
20	20
28	1

Tabla finger del nodo 12

(b)

Figura 5-24. (a) Conjunto de 32 identificadores de nodos ordenados en círculo. Los sombreados corresponden a máquinas reales. Los arcos muestran los fingers de los nodos 1, 4 y 12. Las etiquetas de los arcos son los índices de las tablas. (b) Ejemplos de las tablas finger.

distribuye al azar sobre los nodos. Para tolerancia a fallas, podrían utilizarse *p* funciones de *hash* diferentes para almacenar cada tupla en *p* nodos, pero esto no lo tomaremos en cuenta aquí.

Si posteriormente algún usuario desea buscar *nombre*, le aplica la función de *hash* para obtener *clave* y después utiliza *sucesor*(*clave*) para encontrar la dirección IP del nodo que almacena sus tuplas de índice. El primer paso es fácil; el segundo no lo es. Para poder encontrar la dirección IP del nodo que corresponde a cierta clave, cada nodo debe mantener ciertas estructuras de datos administrativas. Una de ellas es la dirección IP de su nodo sucesor a lo largo del círculo identificador de nodo. Por ejemplo, en la figura 5-24, el sucesor del nodo 4 es 7 y el sucesor del nodo 7 es 12.

La búsqueda ahora puede ser como se muestra a continuación. El nodo solicitante envía un paquete a su sucesor, el cual contiene su dirección IP y la clave que está buscando. El paquete se propaga alrededor del anillo hasta que localiza al sucesor del identificador de nodo que se está buscando. Ese nodo verifica si tiene alguna información que corresponda con la clave y, de ser así, la regresa directamente al nodo solicitante, del cual tiene la dirección IP.

Como primera optimización, cada nodo puede contener las direcciones IP tanto de su sucesor como de su predecesor, por lo que las consultas pueden enviarse en dirección de las manecillas del reloj o en dirección contraria, dependiendo de cuál ruta se considere más corta. Por ejemplo, el nodo 7 de la figura 5-24 puede ir en dirección de las manecillas del reloj para encontrar el identificador de nodo 10, pero en dirección contraria para encontrar el identificador de nodo 3.

Incluso con dos opciones de dirección, la búsqueda lineal de todos los nodos es muy ineficiente en un sistema grande de igual a igual debido a que el número promedio de nodos requeridos por búsqueda es $n/2$. Para incrementar en forma considerable la velocidad de la búsqueda, cada nodo también mantiene lo que Chord llama una **tabla finger.** Ésta tiene m entradas, indexadas desde 0 hasta $m - 1$, y cada una apunta a un nodo real diferente. Cada una de estas entradas tiene dos campos: *inicio* y la dirección IP de *sucesor*(*inicio*), como se muestra para tres nodos de ejemplo de la figura 5-24(b). Los valores de los campos para la entrada i en el nodo k son:

$inicio = k + 2^i \ (módulo 2^m)$
Dirección IP de *sucesor*(*inicio* [*i*])

Observe que cada nodo almacena las direcciones IP de una cantidad de nodos relativamente pequeña y que la mayoría de éstos están muy cercanos en términos de identificador de nodo.

Mediante la tabla *finger*, la búsqueda de *clave* en el nodo k se realiza como se muestra a continuación. Si *clave* está entre k y *sucesor*(k), el nodo que contiene información acerca de *clave* es *sucesor* (k) y la búsqueda termina. De lo contrario, en la tabla *finger* se busca la entrada cuyo campo *inicio* sea el predecesor más cercano de *clave*. A continuación se envía una solicitud directamente a la dirección IP de esa entrada de la tabla *finger* para pedirle que continúe la búsqueda. Debido a que dicha dirección está más cerca de la *clave,* aunque todavía está por debajo, es muy probable que pueda regresar la respuesta con tan sólo algunas consultas adicionales. De hecho, debido a que cada búsqueda divide en dos la distancia restante al destino, puede mostrarse que el número promedio de búsquedas es $\log_2 n$.

Como primer ejemplo, considere la búsqueda de *clave* = 3 en el nodo 1. Debido a que el nodo 1 sabe que 3 está entre él y su sucesor, 4, el nodo deseado es 4 y la búsqueda termina, regresando la dirección IP del nodo 4.

Como segundo ejemplo, considere la búsqueda de *clave* = 14 en el nodo 1. Debido a que 14 no está entre 1 y 4, se consulta la tabla *finger*. El predecesor más cercano a 14 es 9, por lo que la solicitud se reenvía a la dirección IP de la entrada 9, es decir, la del nodo 12. Este nodo ve que 14 está entre él y su sucesor (15), por lo que regresa la dirección IP del nodo 15.

Como tercer ejemplo, considere la búsqueda de *clave* = 16 en el nodo 1. Nuevamente, se envía una solicitud al nodo 12, pero esta vez dicho nodo no sabe la respuesta. Busca el nodo más cercano que preceda a 16 y encuentra 14, lo que resulta en la dirección IP del nodo 15. A continuación

se envía una consulta ahí. El nodo 15 observa que 16 está entre él y su sucesor (20), por lo que regresa la dirección IP de 20 al invocador, que en este caso es el nodo 1.

Puesto que los nodos se unen y separan todo el tiempo, Chord necesita una forma de manejar estas operaciones. Suponemos que cuando el sistema comenzó a operar era tan pequeño que los nodos apenas podían intercambiar información directamente para construir el primer círculo y las primeras tablas *finger*. Después de eso se necesita un procedimiento automatizado, como el que se muestra a continuación. Cuando un nuevo nodo, *r*, desea unirse, debe contactar a algún nodo existente y pedirle que busque la dirección IP de *sucesor*(*r*). A continuación, el nuevo nodo solicita a *sucesor*(*r*) su predecesor. Después pide a ambos que inserten *r* entre ellos en el círculo. Por ejemplo, si el nodo 24 de la figura 5-24 desea unirse, pide a cualquier nodo que busque *sucesor*(24), que es 27. A continuación pide a 27 su predecesor (20). Después de que les informa a estos dos de su existencia, 20 utiliza 24 como su sucesor y 27 utiliza 24 como su predecesor. Además, el nodo 27 entrega esas claves en el rango 21–24, que ahora pertenece a 24. En este punto, 24 está completamente insertado.

Sin embargo, ahora muchas tablas *finger* son erróneas. Para corregirlas, cada nodo ejecuta un proceso en segundo plano que recalcula de manera periódica cada *finger* invocando a *sucesor*. Cuando una de estas consultas coincide con un nuevo nodo, se actualiza la entrada correspondiente del *finger*.

Cuando un nodo se separa con éxito, entrega sus claves a su sucesor e informa a su predecesor su partida a fin de que dicho predecesor pueda enlazarse con el sucesor del nodo que se va. Cuando falla un nodo, surge un problema pues su predecesor ya no tiene un sucesor válido. Para solucionarlo, cada nodo lleva un registro no sólo de su sucesor directo sino también de sus s sucesores directos, a fin de saltar hasta $s - 1$ nodos erróneos consecutivos y volver a conectar el círculo.

Chord se ha utilizado para construir un sistema de archivos distribuido (Dabek y cols., 2001b) y otras aplicaciones, y las investigaciones continúan. En (Rowstron y Druschel, 2001a, y Rowstron y Druschel, 2001b), se describe un sistema de igual a igual diferente, Pastry, así como sus aplicaciones. En (Clarke y cols., 2002) se analiza un tercer sistema de igual a igual, Freenet. En (Ratnasamy y cols., 2001) se describe un cuarto sistema de este tipo.

5.3 ALGORITMOS DE CONTROL DE CONGESTIÓN

Cuando hay demasiados paquetes presentes en la subred (o en una parte de ella), hay una degradación del desempeño. Esta situación se llama **congestión.** En la figura 5-25 se muestra este síntoma. Cuando la cantidad de paquetes descargados en la subred por los *hosts* está dentro de su capacidad de conducción, todos se entregan (excepto unos pocos afligidos por errores de transmisión) y la cantidad entregada es proporcional al número enviado. Sin embargo, a medida que aumenta el tráfico, los enrutadores ya no pueden manejarlo y comienzan a perder paquetes. Esto tiende a empeorar las cosas. Con mucho tráfico, el desempeño se desploma por completo y casi no hay entrega de paquetes.

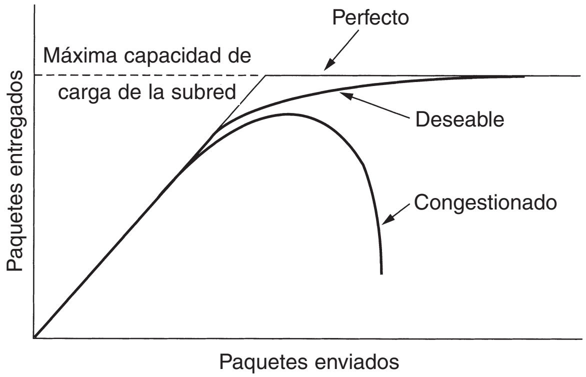


Figura 5-25. Cuando se genera demasiado tráfico, ocurre congestión y se degrada marcadamente el desempeño.

La congestión puede ocurrir por varias razones. Si de manera repentina comienzan a llegar cadenas de paquetes por tres o cuatro líneas de entrada y todas necesitan la misma línea de salida, se generará una cola. Si no hay suficiente memoria para almacenar a todos los paquetes, algunos de ellos se perderán. La adición de memoria puede ayudar hasta cierto punto, pero Nagle (1987) descubrió que si los enrutadores tienen una cantidad infinita de memoria, la congestión empeora en lugar de mejorar, ya que para cuando los paquetes llegan al principio de la cola, su temporizador ha terminado (repetidamente) y se han enviado duplicados. Todos estos paquetes serán debidamente reenviados al siguiente enrutador, aumentando la carga en todo el camino hasta el destino.

Los procesadores lentos también pueden causar congestión. Si las CPUs de los enrutadores son lentas para llevar a cabo las tareas de administración requeridas (búferes de encolamiento, actualización de tablas, etcétera), las colas pueden alargarse, aun cuando haya un exceso de capacidad de línea. De la misma manera, las líneas de poco ancho de banda también pueden causar congestión. La actualización de las líneas sin cambiar los procesadores, o viceversa, por lo general ayuda un poco, pero con frecuencia simplemente desplaza el cuello de botella. Además, actualizar sólo parte de un sistema simplemente mueve el cuello de botella a otra parte. El problema real con frecuencia es un desajuste entre partes del sistema. Este problema persistirá hasta que todos los componentes estén en equilibrio.

Vale la pena indicar de manera explícita la diferencia entre el control de la congestión y el control de flujo, pues la relación es sutil. El control de congestión se ocupa de asegurar que la subred sea capaz de transportar el tráfico ofrecido. Es un asunto global, en el que interviene el comportamiento de todos los *hosts*, todos los enrutadores, el proceso de almacenamiento y reenvío dentro de los enrutadores y todos los demás factores que tienden a disminuir la capacidad de transporte de la subred.

En contraste, el control de flujo se relaciona con el tráfico punto a punto entre un emisor dado y un receptor dado. Su tarea es asegurar que un emisor rápido no pueda transmitir datos de manera continua a una velocidad mayor que la que puede absorber el receptor. El control de flujo casi

siempre implica una retroalimentación directa del receptor al emisor, para indicar al emisor cómo van las cosas en el otro lado.

Para captar la diferencia entre estos dos conceptos, considere una red de fibra óptica con una capacidad de 1000 gigabits/seg en la que una supercomputadora está tratando de transferir un archivo a una computadora personal que opera a 1 Gbps. Aunque no hay congestión (la red misma no está en problemas), se requiere control de flujo para obligar a la supercomputadora a detenerse con frecuencia para darle a la computadora personal un momento de respiro.

En el otro extremo, considere una red de almacenamiento y reenvío con líneas de 1 Mbps y 1000 computadoras grandes, la mitad de las cuales trata de transferir archivos a 100 kbps a la otra mitad. Aquí el problema no es que los emisores rápidos saturen a los receptores lentos, sino simplemente que el tráfico ofrecido total excede lo que la red puede manejar.

La razón por la que se confunden con frecuencia el control de congestión y el control de flujo es que algunos algoritmos de control de congestión operan enviando mensajes de regreso a varios orígenes, indicándoles que reduzcan su velocidad cuando la red se mete en problemas. Por lo tanto, un *host* puede recibir un mensaje de "reducción de velocidad" porque el receptor no puede manejar la carga o porque la red no la puede manejar. Más adelante regresaremos a este punto.

Comenzaremos nuestro estudio del control de congestión examinando un modelo general para manejarlo. Luego veremos métodos generales para prevenirlo. Después de eso, veremos varios algoritmos dinámicos para manejarlo una vez que se ha establecido.

5.3.1 Principios generales del control de congestión

Muchos problemas de los sistemas complejos, como las redes de computadoras, pueden analizarse desde el punto de vista de una teoría de control. Este método conduce a dividir en dos grupos todas las soluciones: de ciclo abierto y de ciclo cerrado. En esencia, las soluciones de ciclo abierto intentan resolver el problema mediante un buen diseño, para asegurarse en primer lugar de que no ocurra. Una vez que el sistema está en funcionamiento, no se hacen correcciones a medio camino.

Las herramientas para llevar a cabo control de ciclo abierto incluyen decidir cuándo aceptar tráfico nuevo, decidir cuándo descartar paquetes, y cuáles, y tomar decisiones de calendarización en varios puntos de la red. Todas tienen en común el hecho de que toman decisiones independientemente del estado actual de la red.

En contraste, las soluciones de ciclo cerrado se basan en el concepto de un ciclo de retroalimentación. Este método tiene tres partes cuando se aplica al control de congestión:

1. Monitorear el sistema para detectar cuándo y dónde ocurren congestiones.
2. Pasar esta información a lugares en los que puedan llevarse a cabo acciones.
3. Ajustar la operación del sistema para corregir el problema.

Es posible utilizar varias métricas para monitorear la subred en busca de congestiones. Las principales son el porcentaje de paquetes descartados debido a falta de espacio de búfer, la longitud

promedio de las colas, la cantidad de paquetes para los cuales termina el temporizador y se transmiten de nueva cuenta, el retardo promedio de los paquetes y la desviación estándar del retardo de paquete. En todos los casos, un aumento en las cifras indica un aumento en la congestión.

El segundo paso del ciclo de retroalimentación es la transferencia de información relativa a la congestión desde el punto en que se detecta hasta el punto en que puede hacerse algo al respecto. La manera más obvia es que el enrutador que detecta la congestión envíe un paquete al origen (u orígenes) del tráfico, anunciando el problema. Por supuesto, estos paquetes adicionales aumentan la carga precisamente en el momento en que no se necesita más carga, es decir, cuando la subred está congestionada.

Por fortuna, existen otras opciones. Por ejemplo, en cada paquete puede reservarse un bit o campo para que los enrutadores lo llenen cuando la congestión rebase algún umbral. Cuando un enrutador detecta este estado congestionado, llena el campo de todos los paquetes de salida, para avisar a los vecinos.

Otra estrategia es hacer que los *hosts* o enrutadores envíen de manera periódica paquetes de sondeo para preguntar explícitamente sobre la congestión. Esta información puede usarse para enrutar el tráfico fuera de áreas con problemas. Algunas estaciones de radio tienen helicópteros que vuelan sobre la ciudad para informar de la congestión en las calles, con la esperanza de que los escuchas enrutarán sus paquetes (autos) fuera de las zonas conflictivas.

En todos los esquemas de retroalimentación, la esperanza es que el conocimiento sobre la congestión hará que los *hosts* emprendan acciones adecuadas con miras a reducir la congestión. Para operar en forma correcta, la escala de tiempo debe ajustarse con cuidado. Si el enrutador grita ALTO cada vez que llegan dos paquetes seguidos, y SIGA, cada vez que está inactivo durante 20 μseg, el sistema oscilará sin control y nunca convergerá. Por otra parte, si un enrutador espera 30 minutos para asegurarse antes de tomar una decisión, el mecanismo de control de congestión reaccionará tan lentamente que no será de utilidad. Para funcionar bien se requiere un justo medio, pero encontrar la constante de tiempo correcta no es un asunto trivial.

Se conocen muchos algoritmos de control de congestión. A fin de organizarlos lógicamente, Yang y Reddy (1995) han desarrollado una taxonomía de los algoritmos de control de congestión. Comienzan por dividir todos los algoritmos en ciclo abierto y ciclo cerrado, como se describió anteriormente. Dividen todavía más los algoritmos de ciclo abierto en algoritmos que actúan en el origen y los que actúan en el destino. Los algoritmos de ciclo cerrado también se dividen en dos subcategorías: retroalimentación explícita e implícita. En los algoritmos de retroalimentación explícita, regresan paquetes desde el punto de congestión para avisar al origen. En los algoritmos implícitos, el origen deduce la existencia de una congestión haciendo observaciones locales, como el tiempo necesario para que regresen las confirmaciones de recepción.

La presencia de congestión significa que la carga es (temporalmente) superior (en una parte del sistema) a la que pueden manejar los recursos. Vienen a la mente dos soluciones: aumentar los recursos o disminuir la carga. Por ejemplo, la subred puede comenzar a utilizar líneas de acceso telefónico para aumentar de manera temporal el ancho de banda entre ciertos puntos. En los sistemas satelitales la potencia de transmisión creciente con frecuencia reditúa un ancho de banda más alto. La división del tráfico entre varias rutas en lugar de usar siempre la mejor también aumenta efectivamente el ancho de banda. Por último, a fin de contar con mayor capacidad, los enrutadores

de repuesto que normalmente sirven sólo como respaldo (para hacer que el sistema sea tolerante a fallas), pueden ponerse en línea cuando aparece una congestión severa.

Sin embargo, a veces no es posible aumentar la capacidad, o ya ha sido aumentada al máximo. Entonces, la única forma de combatir la congestión es disminuir la carga. Existen varias maneras de reducir la carga, como negar el servicio a algunos usuarios, degradar el servicio para algunos o todos los usuarios y obligar a los usuarios a programar sus solicitudes de una manera más predecible.

Algunos de estos métodos, que estudiaremos en breve, se aplican mejor a los circuitos virtuales. En las subredes que usan circuitos virtuales de manera interna, estos métodos pueden usarse en la capa de red. En las subredes de datagramas algunas veces se pueden utilizar en las conexiones de capa de transporte. En este capítulo nos enfocaremos en su uso en la capa de red. En el siguiente veremos lo que puede hacerse en la capa de transporte para manejar la congestión.

5.3.2 Políticas de prevención de congestión

Comencemos nuestro estudio de los métodos de control de congestión estudiando los sistemas de ciclo abierto. Estos sistemas están diseñados para reducir al mínimo la congestión desde el inicio, en lugar de permitir que ocurra y reaccionar después del hecho. Tratan de lograr su objetivo usando políticas adecuadas en varios niveles. En la figura 5-26 vemos diferentes políticas para las capas de enlace de datos, red y transporte que pueden afectar a la congestión (Jain, 1990).

Capa	Políticas
Transporte	• Política de retransmisión • Política de almacenamiento en caché de paquetes fuera de orden • Política de confirmaciones de recepción • Política de control de flujo • Determinación de terminaciones de temporizador
Red	• Circuitos virtuales vs. datagramas en la subred • Política de encolamiento y servicio de paquetes • Política de descarte de paquetes • Algoritmo de enrutamiento • Administración de tiempo de vida del paquete
Enlace de datos	• Política de retransmisiones • Política de almacenamiento en caché de paquetes fuera de orden • Política de confirmación de recepción • Política de control de flujo

Figura 5-26. Políticas relacionadas con la congestión.

Comencemos por la capa de enlace de datos y avancemos hacia arriba. La política de retransmisiones tiene que ver con la rapidez con la que un emisor termina de temporizar y con lo que transmite al ocurrir una terminación de temporizador. Un emisor nervioso que a veces termina de temporizar demasiado pronto y retransmite todos los paquetes pendientes usando el protocolo de

retroceso n impondrá una carga más pesada al sistema que un emisor calmado que usa repetición selectiva. La política de almacenamiento en caché está muy relacionada con esto. Si los receptores descartan de manera rutinaria todos los paquetes que llegan fuera de orden, posteriormente se tendrán que enviar otra vez, lo que creará una carga extra. Con respecto al control de congestión, la repetición selectiva es mejor que el retroceso n.

La política de confirmación de recepción también afecta la congestión. Si la recepción de cada paquete se confirma de inmediato, los paquetes de confirmación de recepción generan tráfico extra. Sin embargo, si se guardan las confirmaciones de recepción para sobreponerlas en el tráfico en sentido inverso, pueden resultar en terminaciones de temporizador y retransmisiones extra. Un esquema de control de flujo estricto (por ejemplo, una ventana pequeña) reduce la tasa de datos y permite, por lo tanto, atacar la congestión.

En la capa de red, la decisión entre circuitos virtuales y datagramas afecta la congestión, ya que muchos algoritmos de control de congestión sólo funcionan con subredes de circuitos virtuales. La política de encolamiento y servicio de paquetes se refiere a que los enrutadores tengan una cola por línea de entrada, y una o varias colas por línea de salida. También se relaciona con el orden en que se procesan los paquetes (por ejemplo, en *round robin* o con base en prioridades). La política de descarte es la regla que indica qué paquete se descarta cuando no hay espacio. Una buena política puede ayudar a aliviar la congestión y una mala puede hacerlo peor.

Un buen algoritmo de enrutamiento puede evitar la congestión si distribuye el tráfico entre todas las líneas, pero uno malo puede enviar demasiado tráfico por líneas ya congestionadas. Por último, la administración del tiempo de vida de los paquetes se encarga del tiempo que puede existir un paquete antes de ser descartado. Si este tiempo es demasiado grande, los paquetes perdidos pueden bloquear la operación durante un buen rato, pero si es demasiado corto, los paquetes pueden expirar antes de llegar a su destino, lo que provoca retransmisiones.

En la capa de transporte surgen los mismos problemas que en la capa de enlace de datos, pero además es más difícil la determinación del intervalo de expiración, porque el tiempo de tránsito a través de la red es menos predecible que el tiempo de tránsito por un cable entre dos enrutadores. Si el intervalo es demasiado corto, se enviarán paquetes extra de manera innecesaria. Si es muy largo, se reducirá la congestión, pero el tiempo de respuesta se verá afectado cada vez que se pierda un paquete.

5.3.3 Control de congestión en subredes de circuitos virtuales

Los métodos de control de congestión antes descritos son básicamente de ciclo abierto: tratan de evitar que ocurran las congestiones, en lugar de manejarlas una vez ocurridas. En esta sección describiremos algunos métodos para el control dinámico de la congestión en las subredes de circuitos virtuales. En las siguientes dos veremos técnicas que pueden usarse en cualquier subred.

Una de las técnicas que se usa ampliamente para evitar que empeoren las congestiones que ya han comenzado es el **control de admisión.** La idea es sencilla: una vez que se ha detectado la congestión, no se establecen circuitos virtuales nuevos hasta que ha desaparecido el problema. Por

lo tanto, fallan los intentos por establecer conexiones nuevas de capa de transporte. Permitir el acceso a más personas simplemente empeoraría las cosas. Aunque este métodos es simple, su implementación es sencilla. En el sistema telefónico, cuando un conmutador se sobrecarga también se pone en práctica el control de admisión, al no darse tonos de marcado.

Un método alterno es permitir el establecimiento de nuevos circuitos virtuales, pero enrutando cuidadosamente los circuitos nuevos por otras rutas que no tengan problemas. Por ejemplo, considere la subred de la figura 5-27(a), en la que dos enrutadores están congestionados.

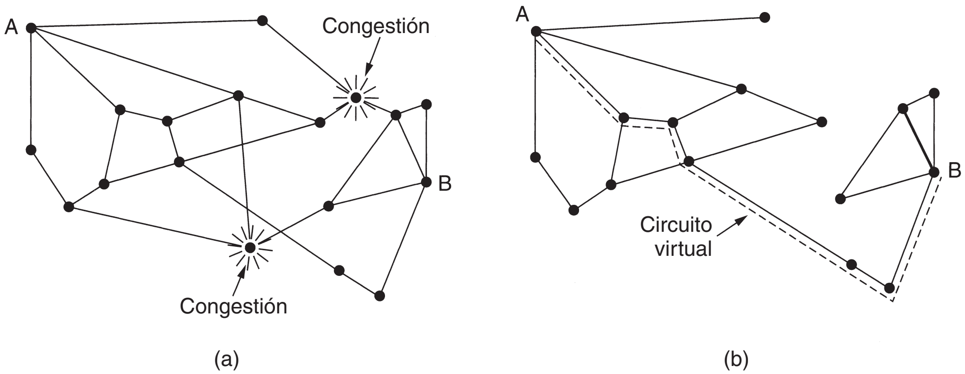


Figura 5-27. (a) Subred congestionada. (b) Subred redibujada que elimina la congestión. También se muestra un circuito virtual de *A* a *B*.

Suponga que un *host* conectado al enrutador *A* quiere establecer una conexión con un *host* conectado al enrutador *B*. Normalmente esta conexión pasaría a través de uno de los enrutadores congestionados. Para evitar esta situación, podemos redibujar la subred como se muestra en la figura 5-27(b), omitiendo los enrutadores congestionados y todas sus líneas. La línea punteada muestra una ruta posible para el circuito virtual que evita los enrutadores congestionados.

Otra estrategia que tiene que ver con los circuitos virtuales es negociar un acuerdo entre el *host* y la subred cuando se establece un circuito virtual. Este arreglo normalmente especifica el volumen y la forma del tráfico, la calidad de servicio requerida y otros parámetros. Para cumplir con su parte del acuerdo, la subred por lo general reservará recursos a lo largo de la ruta cuando se establezca el circuito. Estos recursos pueden incluir espacio en tablas y en búfer en los enrutadores y ancho de banda en las líneas. De esta manera, es poco probable que ocurran congestiones en los circuitos virtuales nuevos, porque está garantizada la disponibilidad de todos los recursos necesarios.

Este tipo de reservación puede llevarse a cabo todo el tiempo como procedimiento operativo estándar, o sólo cuando la subred está congestionada. Una desventaja de hacerlo todo el tiempo es que se tiende a desperdiciar recursos. Si seis circuitos virtuales que podrían usar 1 Mbps pasan por la misma línea física de 6 Mbps, la línea tiene que marcarse como llena, aunque pocas veces ocurra que los seis circuitos virtuales transmitan a toda velocidad al mismo tiempo. En consecuencia, el precio del control de congestión es un ancho de banda sin utilizar (es decir, desperdiciado).

5.3.4 Control de congestión en subredes de datagramas

Veamos ahora un enfoque que puede usarse en subredes de datagramas (y también en subredes de circuitos virtuales). Cada enrutador puede monitorear fácilmente el uso de sus líneas de salida y de otros recursos. Por ejemplo, puede asociar cada línea a una variable real, u, cuyo valor, entre 0.0 y 1.0, refleja el uso reciente de esa línea. Para tener una buena estimación de u, puede tomarse periódicamente una muestra del uso instantáneo de la línea, f (0 o 1), y actualizar u periódicamente de acuerdo con

$$u_{\text{nvo}} = au_{\text{ant}} + (1 - a)f$$

donde la constante a determina la rapidez con que el enrutador olvida la historia reciente.

Siempre que u rebasa el umbral, la línea de salida entra en un estado de "advertencia". Cada paquete nuevo que llega se revisa para ver si su línea de salida está en el estado de advertencia. Si es así, se realiza alguna acción. Ésta puede ser una de varias alternativas, que analizaremos a continuación.

El bit de advertencia

La arquitectura DECNET antigua señalaba el estado de advertencia activando un bit especial en el encabezado del paquete. Frame relay también lo hace. Cuando el paquete llegaba a su destino, la entidad transportadora copiaba el bit en la siguiente confirmación de recepción que se regresaba al origen. A continuación el origen reducía el tráfico.

Mientras el enrutador estuviera en el estado de advertencia, continuaba activando el bit de advertencia, lo que significaba que el origen continuaba obteniendo confirmaciones de recepción con dicho bit activado. El origen monitoreaba la fracción de confirmaciones de recepción con el bit activado y ajustaba su tasa de transmisión de manera acorde. En tanto los bits de advertencia continuaran fluyendo, el origen continuaba disminuyendo su tasa de transmisión. Cuando disminuía lo suficiente, el origen incrementaba su tasa de transmisión. Observe que debido a que cada enrutador a lo largo de la ruta podía activar el bit de advertencia, el tráfico se incrementaba sólo cuando no había enrutadores con problemas.

Paquetes reguladores

El algoritmo de control de congestión anterior es muy sutil. Utiliza medios indirectos para indicar al origen que vaya más despacio. ¿Por qué no indicárselo de manera directa? En este método, el enrutador regresa un **paquete regulador** al *host* de origen, proporcionándole el destino encontrado en el paquete. El paquete original se etiqueta (se activa un bit del encabezado) de manera que no genere más paquetes reguladores más adelante en la ruta y después se reenvía de la manera usual.

Cuando el *host* de origen obtiene el paquete regulador, se le pide que reduzca en un porcentaje X el tráfico enviado al destino especificado. Puesto que otros paquetes dirigidos al mismo destino probablemente ya están en camino y generarán más paquetes reguladores, el *host* debe ignorar

los paquetes reguladores que se refieran a ese destino por un intervalo fijo de tiempo. Una vez que haya expirado ese tiempo, el *host* escucha más paquetes reguladores durante otro intervalo. Si llega alguno, la línea todavía está congestionada, por lo que el *host* reduce el flujo aún más y comienza a ignorar nuevamente los paquetes reguladores. Si no llega ningún paquete de este tipo durante el periodo de escucha, el *host* puede incrementar el flujo otra vez. La retroalimentación implícita de este protocolo puede ayudar a evitar la congestión que aún no estrangula ningún flujo a menos que ocurra un problema.

Los *hosts* pueden reducir el tráfico ajustando los parámetros de sus políticas, por ejemplo, su tamaño de ventana. Por lo general, el primer paquete regulador causa que la tasa de datos se reduzca en 0.50 con respecto a su tasa anterior, el siguiente causa una reducción de 0.25, etcétera. Los incrementos se dan en aumentos más pequeños para evitar que la congestión se vuelva a generar rápidamente.

Se han propuesto algunas variaciones de este algoritmo de control de congestión. En una, los enrutadores pueden mantener varios umbrales. Dependiendo de qué umbral se ha rebasado, el paquete regulador puede contener una advertencia suave, una severa o un ultimátum.

Otra variación es utilizar como señal de activación tamaños de colas o utilización de los búferes en lugar de la utilización de la línea. Es posible utilizar la misma ponderación exponencial con esta métrica como con *u*, por supuesto.

Paquetes reguladores de salto por salto

A altas velocidades o distancias grandes, el envío de un paquete regulador a los *hosts* de origen no funciona bien porque la reacción es muy lenta. Por ejemplo, considere a un *host* en San Francisco (enrutador *A* de la figura 5-28) que está enviando tráfico a un *host* en Nueva York (enrutador *D* de la figura 5-28) a 155 Mbps. Si al *host* de Nueva York se le comienza a terminar el espacio de búfer, un paquete regulador tardará unos 30 mseg en regresar a San Francisco para indicarle que disminuya su velocidad. La propagación de paquetes reguladores se muestra en el segundo, tercer y cuarto pasos de la figura 5-28(a). En esos 30 mseg se habrán enviado otros 4.6 megabits. Aun si el *host* de San Francisco se desactiva de inmediato, los 4.6 megabits en la línea continuarán fluyendo, y habrá que encargarse de ellos. Sólo hasta el séptimo diagrama de la figura 5-28(a) el enrutador de Nueva York notará un flujo más lento.

Un método alterno es hacer que el paquete regulador ejerza su efecto en cada salto que dé, como se muestra en la secuencia de la figura 5-28(b). Aquí, una vez que el paquete regulador llega a *F*, se obliga a *F* a reducir el flujo a *D*. Hacerlo requerirá que *F* destine más búferes al flujo, ya que el origen aún está transmitiendo a toda velocidad, pero da a *D* un alivio inmediato, como en un mensaje comercial de un remedio contra el dolor de cabeza. En el siguiente paso, el paquete regulador llega a *E*, e indica a éste que reduzca el flujo a *F*. Esta acción impone una mayor carga a los búferes de *E*, pero da un alivio inmediato a *F*. Por último, el paquete regulador llega a *A* y efectivamente se reduce el flujo.

El efecto neto de este esquema de salto por salto es proporcionar un alivio rápido al punto de congestión, a expensas de usar más búferes ascendentes. De esta manera puede cortarse la congestión

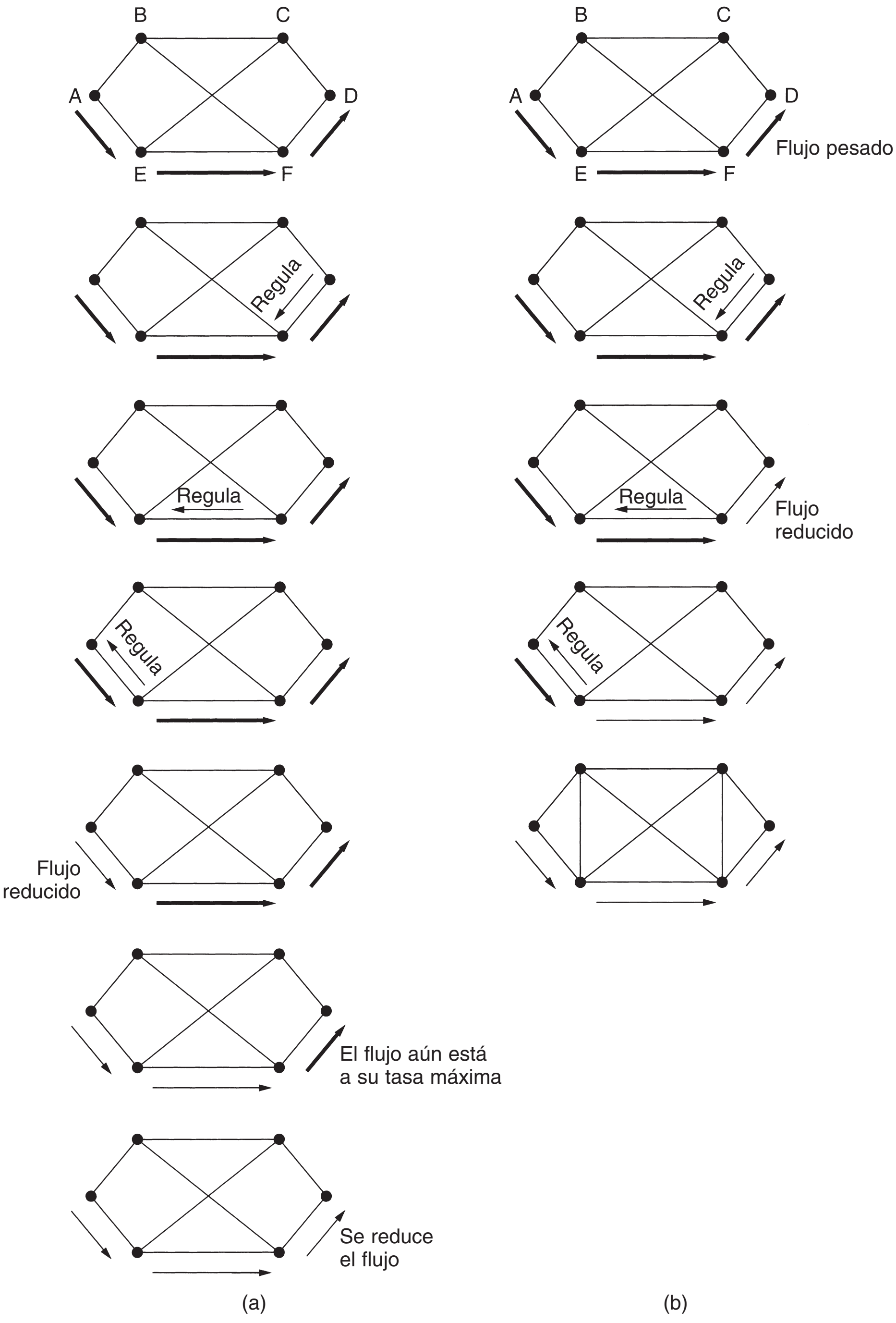


Figura 5-28. (a) Paquete regulador que afecta sólo al origen. (b) Paquete regulador que afecta cada salto por el que pasa.

en la raíz, sin que se pierdan paquetes. La idea se estudia con mayor detalle en Mishra y Kanakia, 1992.

5.3.5 Desprendimiento de carga

Cuando ninguno de los métodos anteriores elimina la congestión, los enrutadores pueden sacar la artillería pesada: el **desprendimiento de carga,** que es una manera rebuscada de decir que, cuando se inunda a los enrutadores con paquetes que no pueden manejar, simplemente los tiran. El término viene del mundo de la generación de energía eléctrica, donde se refiere a la práctica de instalaciones que intencionalmente producen apagones en ciertas áreas para salvar a la red completa de venirse abajo en días calurosos de verano en los que la demanda de energía eléctrica excede por mucho el suministro.

Un enrutador abrumado por paquetes simplemente puede escoger paquetes al azar para desprenderse de ellos, pero normalmente puede hacer algo mejor. El paquete a descartar puede depender de las aplicaciones que se estén ejecutando. En la transferencia de archivos vale más un paquete viejo que uno nuevo, pues el deshacerse del paquete 6 y mantener los paquetes 7 a 10 causará un hueco en el receptor que podría obligar a que se retransmitan los paquetes 6 a 10 (si el receptor descarta de manera rutinaria los paquetes en desorden). En un archivo de 12 paquetes, deshacerse del paquete 6 podría requerir la retransmisión de los paquetes 7 a 12, y deshacerse del 10 podría requerir la retransmisión sólo del 10 al 12. En contraste, en multimedia es más importante un paquete nuevo que uno viejo. La política anterior (más viejo es mejor que más nuevo), con frecuencia se llama **vino,** y la última (más nuevo es mejor que más viejo) con frecuencia se llama **leche.**

Un paso adelante de esto en cuanto a inteligencia requiere la cooperación de los emisores. En muchas aplicaciones, algunos paquetes son más importantes que otros. Por ejemplo, ciertos algoritmos de compresión de vídeo transmiten periódicamente una trama entera y sus tramas subsiguientes como diferencias respecto a la última trama completa. En este caso, es preferible desprenderse de un paquete que es parte de una diferencia que desprenderse de uno que es parte de una trama completa. Como otro ejemplo, considere la transmisión de un documento que contiene texto ASCII e imágenes. La pérdida de una línea de píxeles de una imagen es mucho menos dañina que la pérdida de una línea de texto legible.

Para poner en práctica una política inteligente de descarte, las aplicaciones deben marcar sus paquetes con clases de prioridades para indicar su importancia. Si lo hacen, al tener que descartar paquetes, los enrutadores pueden descartar primero los paquetes de clase más baja, luego los de la siguiente clase más baja, etcétera. Por supuesto, a menos que haya una razón poderosa para marcar los paquetes como MUY IMPORTANTE–NUNCA DESCARTAR, nadie lo hará.

La razón podría ser monetaria, siendo más barato el envío de paquetes de baja prioridad que el de los de alta prioridad. Como alternativa, los emisores podrían tener permitido enviar paquetes de alta prioridad bajo condiciones de carga ligera, pero a medida que aumente la carga, los paquetes podrían descartarse, lo que haría que los usuarios ya no siguieran enviándolos.

Otra opción es permitir que los *hosts* excedan los límites especificados en el acuerdo negociado al establecer el circuito virtual (por ejemplo, usar un ancho de banda mayor que el permitido),

pero sujetos a la condición de que el exceso de tráfico se marque con prioridad baja. Tal estrategia de hecho no es mala idea, porque utiliza con mayor eficiencia los recursos inactivos, permitiendo que los *hosts* los utilicen siempre y cuando nadie más esté interesado, pero sin establecer un derecho sobre ellos cuando los tiempos se vuelven difíciles.

Detección temprana aleatoria

Es bien sabido que tratar con la congestión después de que se detecta por primera vez es más efectivo que dejar que dañe el trabajo y luego tratar de solucionarlo. Esta observación conduce a la idea de descartar paquetes antes de que se ocupe todo el espacio de búfer. Un algoritmo popular para realizar esto se conoce como **RED** (**detección temprana aleatoria**) (Floyd y Jacobson, 1993). En algunos protocolos de transporte (entre ellos TCP), la respuesta a paquetes perdidos es que el origen disminuya su velocidad. El razonamiento detrás de esta lógica es que TCP fue diseñado para redes cableadas, y éstas son muy confiables, por lo tanto, la pérdida de paquetes se debe principalmente a desbordamientos de búfer y no a errores de transmisiones. Este hecho puede aprovecharse para reducir la congestión.

El objetivo de hacer que los enrutadores se deshagan de los paquetes antes de que la situación sea irremediable (de aquí el término "temprana" en el nombre), es que haya tiempo para hacer algo antes de que sea demasiado tarde. Para determinar cuándo comenzar a descartarlos, los enrutadores mantienen un promedio móvil de sus longitudes de cola. Cuando la longitud de cola promedio en algunas líneas sobrepasa un umbral, se dice que la línea está congestionada y se toma alguna medida.

Debido a que tal vez el enrutador no puede saber cuál origen está causando la mayoría de los problemas, probablemente lo mejor que se puede hacer es elegir un paquete al azar de la cola que puso en marcha la acción.

¿Cómo puede el enrutador informar al origen sobre el problema? Una forma es enviarle un paquete regulador, como describimos anteriormente. Sin embargo, con ese método surge un problema ya que coloca todavía más carga en la ya congestionada red. Una estrategia diferente es descartar el paquete seleccionado y no reportarlo. El origen notará en algún momento la falta de confirmación de recepción y tomará medidas. Debido a que sabe que los paquetes perdidos por lo general son causados por la congestión y las eliminaciones, responderá reduciendo la velocidad en lugar de aumentarla. Esta forma implícita de retroalimentación sólo funciona cuando los orígenes responden a la pérdida de paquetes reduciendo su tasa de transmisión. En las redes inalámbricas, en las que la mayoría de las pérdidas se debe al ruido en el enlace de radio, no se puede utilizar este método.

5.3.6 Control de fluctuación

En aplicaciones como la transmisión continua de audio y vídeo no importa gran cosa si los paquetes tardan 20 o 30 mseg en ser entregados, siempre y cuando el tiempo de tránsito (retardo) sea constante. La variación (es decir, la desviación estándar) en el retardo de los paquetes se conoce como **fluctuación.** Una fluctuación alta, por ejemplo cuando unos paquetes tardan en llegar a su

destino 20 mseg y otros 30 mseg, resultará en una calidad desigual del sonido o la imagen. En la figura 5-29 se ilustra la fluctuación. Por tanto, el arreglo de que el 99% de los paquetes se entregue con un retardo que esté entre 24.5 y 25.5 mseg puede ser aceptable.

Por supuesto, el rango escogido debe ser factible. Debe tomar en cuenta el retardo causado por la velocidad de la luz y el retardo mínimo a través de los enrutadores y tal vez dejar un periodo corto para algunos retardos inevitables.

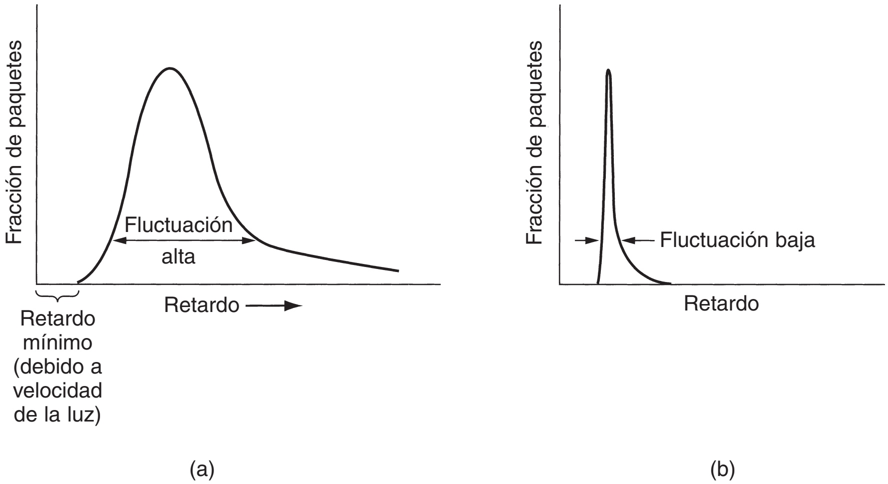


Figura 5-29. (a) Fluctuación alta. (b) Fluctuación baja.

La fluctuación puede limitarse calculando el tiempo de tránsito esperado para cada salto en la ruta. Cuando un paquete llega a un enrutador, éste lo examina para saber qué tan adelantado o retrasado está respecto a lo programado. Esta información se almacena en el paquete y se actualiza en cada salto. Si el paquete está adelantado, se retiene durante el tiempo suficiente para regresarlo a lo programado; si está retrasado, el enrutador trata de sacarlo rápidamente.

De hecho, el algoritmo para determinar cuál de varios paquetes que compiten por una línea de salida debe seguir siempre puede escoger el paquete más retrasado. De esta manera, los paquetes adelantados se frenan y los retrasados se aceleran, reduciendo en ambos casos la cantidad de fluctuación.

En algunas aplicaciones, como el vídeo bajo demanda, la fluctuación puede eliminarse almacenando los datos en el búfer del receptor y después obteniéndolos de dicho búfer en lugar de utilizar la red en tiempo real. Sin embargo, para otras aplicaciones, especialmente aquellas que requieren interacción en tiempo real entre personas como la telefonía y videoconferencia en Internet, el retardo inherente del almacenamiento en el búfer no es aceptable.

El control de congestión es un área activa de investigación. El estado presente se resume en (Gevros y cols., 2001).

5.4 CALIDAD DEL SERVICIO

Las técnicas que observamos en las secciones previas se diseñaron para reducir la congestión y mejorar el rendimiento de la red. Sin embargo, con el crecimiento de las redes de multimedia, con frecuencia estas medidas *ad hoc* no son suficientes. Se necesitan intentos serios para garantizar la calidad del servicio a través del diseño de redes y protocolos. En las siguientes secciones continuaremos nuestro estudio del rendimiento de la red, pero por ahora nos enfocaremos en las formas de proporcionar una calidad de servicio que se ajuste a las necesidades de las aplicaciones. Sin embargo, se debe dejar claro desde el principio que muchas de estas ideas están empezando y sujetas a cambios.

5.4.1 Requerimientos

Un **flujo** es un conjunto de paquetes que van de un origen a un destino. En una red orientada a la conexión, todos los paquetes que pertenezcan a un flujo siguen la misma ruta; en una red sin conexión, pueden seguir diferentes rutas. La necesidad de cada flujo se puede caracterizar por cuatro parámetros principales: confiabilidad, retardo, fluctuación y ancho de banda. Estos parámetros en conjunto determinan la **QoS** (**calidad del servicio**) que el flujo requiere. En la figura 5-30 se listan varias aplicaciones y el nivel de sus requerimientos.

Aplicación	Confiabilidad	Retardo	Fluctuación	Ancho de banda
Correo electrónico	Alta	Bajo	Baja	Bajo
Transferencia de archivos	Alta	Bajo	Baja	Medio
Acceso a Web	Alta	Medio	Baja	Medio
Inicio de sesión remoto	Alta	Medio	Media	Bajo
Audio bajo demanda	Baja	Bajo	Alta	Medio
Vídeo bajo demanda	Baja	Bajo	Alta	Alto
Telefonía	Baja	Alto	Alta	Bajo
Videoconferencia	Baja	Alto	Alta	Alto

Figura 5-30. Qué tan rigurosos son los requerimientos de calidad del servicio.

Las primeras cuatro aplicaciones tienen requerimientos rigurosos en cuanto a confiabilidad. No sería posible enviar bits de manera incorrecta. Este objetivo por lo general se alcanza al realizar una suma de verificación de cada paquete y al verificar dicha suma en el destino. Si se daña un paquete en el tránsito, no se confirma su recepción y se volverá a transmitir posteriormente. Esta estrategia proporciona una alta confiabilidad. Las cuatro aplicaciones finales (audio/vídeo) pueden tolerar errores, por lo que ni se realizan ni comprueban sumas de verificación.

Las aplicaciones de transferencia de archivos, incluyendo correo electrónico y vídeo, no son sensibles al retardo. Si todos los paquetes se retrasan unos segundos de manera uniforme, no hay daño. Las aplicaciones interactivas, como la navegación en Web y el inicio de sesión remoto,

son más sensibles a los retardos. Las aplicaciones en tiempo real, como la telefonía y la videoconferencia, tienen requerimientos estrictos de retardo. Si cada una de las palabras de una llamada telefónica se retrasa exactamente por 2.000 seg, los usuarios hallarán la conexión inaceptable. Por otra parte, la reproducción de archivos de audio o vídeo desde un servidor no requiere un retardo bajo.

Las primeras tres aplicaciones no son sensibles a los paquetes que llegan con intervalos de tiempo irregulares entre ellos. El inicio de sesión remoto es algo sensible a esto, debido a que los caracteres en la pantalla aparecerán en pequeñas ráfagas si una conexión tiene mucha fluctuación. El vídeo y especialmente el audio son en extremo sensibles a la fluctuación. Si un usuario está observando vídeo a través de la red y todos los cuadros se retrasan exactamente 2.000 seg, no hay daño. Pero si el tiempo de transmisión varía de manera aleatoria entre 1 y 2 seg, el resultado sería terrible. En el audio, una fluctuación de incluso unos cuantos milisegundos es claramente audible.

Por último, las aplicaciones difieren en sus anchos de banda; el correo electrónico y el inicio de sesión remoto no necesitan mucho, pero el vídeo en todas sus formas sí lo necesita.

Las redes ATM clasifican los flujos en cuatro categorías amplias con respecto a sus demandas de QoS, como se muestra a continuación:

1. Tasa de bits constante (por ejemplo, telefonía).
2. Tasa de bits variable en tiempo real (por ejemplo, videoconferencia comprimida).
3. Tasa de bits variable no constante (por ejemplo, ver una película a través de Internet).
4. Tasa de bits disponible (por ejemplo, transferencia de archivos).

Estas categorías también son útiles para otros propósitos y otras redes. La tasa de bits constante es un intento por simular un cable al proporcionar un ancho de banda uniforme y un retardo uniforme. La tasa de bits variable ocurre cuando el vídeo está comprimido, algunos cuadros están más comprimidos que otros. Por lo tanto, el envío de un cuadro con muchos detalles podría requerir enviar muchos bits en tanto que el envío de una foto de una pared blanca podría comprimirse muy bien. La tasa de bits disponible es para las aplicaciones, como el correo electrónico, que no son sensibles al retardo o a la fluctuación.

5.4.2 Técnicas para alcanzar buena calidad de servicio

Ahora que sabemos algo sobre los requerimientos de QoS, ¿cómo cumplimos con ellos? Bueno, para empezar, no hay una bola mágica. Ninguna técnica proporciona QoS eficiente y confiable de una manera óptima. En su lugar, se ha desarrollado una variedad de técnicas, con soluciones prácticas que con frecuencia se combinan múltiples técnicas. A continuación examinaremos algunas de las técnicas que los diseñadores de sistemas utilizan para alcanzar la QoS.

Sobreaprovisionamiento

Una solución fácil es proporcionar la suficiente capacidad de enrutador, espacio en búfer y ancho de banda como para que los paquetes fluyan con facilidad. El problema con esta solución es que

es costosa. Conforme pasa el tiempo y los diseñadores tienen una mejor idea de cuánto es suficiente, esta técnica puede ser práctica. En cierta medida, el sistema telefónico tiene un sobreaprovisionamiento. Es raro levantar un auricular telefónico y no obtener un tono de marcado instantáneo. Simplemente hay mucha capacidad disponible ahí que la demanda siempre se puede satisfacer.

Almacenamiento en búfer

Los flujos pueden almacenarse en el búfer en el lado receptor antes de ser entregados. Almacenarlos en el búfer no afecta la confiabilidad o el ancho de banda, e incrementa el retardo, pero atenúa la fluctuación. Para el vídeo o audio bajo demanda, la fluctuación es el problema principal, por lo tanto, esta técnica es muy útil.

En la figura 5-29 vimos la diferencia entre fluctuación alta y fluctuación baja. En la figura 5-31 vemos un flujo de paquetes que se entregan con una fluctuación considerable. El paquete 1 se envía desde el servidor a $t = 0$ seg y llega al cliente a $t = 1$ seg. El paquete 2 tiene un retardo mayor; tarda 2 seg en llegar. Conforme llegan los paquetes, se almacenan en el búfer en la máquina cliente.

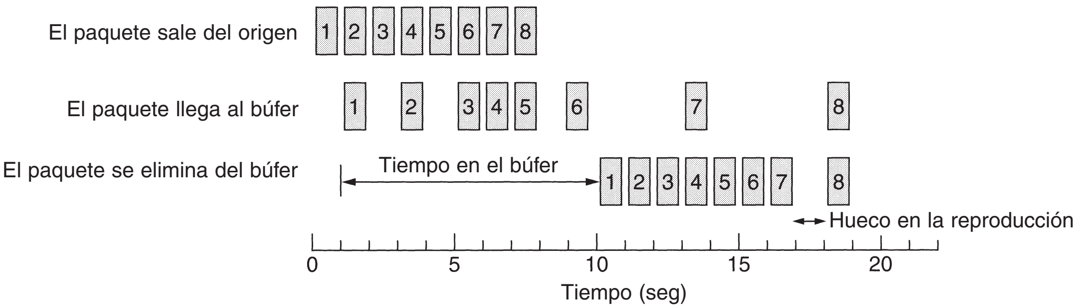


Figura 5-31. Refinamiento del flujo de paquetes almacenándolos en el búfer.

En el seg $t = 10$, la reproducción continúa. En este momento, los paquetes 1 a 6 se han almacenado en el búfer de manera que pueden eliminarse de él en intervalos uniformes para una reproducción suave. Desafortunadamente, el paquete 8 se ha retrasado tanto que no está disponible cuando le toca el turno a su ranura de reproducción, por lo que ésta debe parar hasta que llegue dicho paquete, creando un molesto hueco en la música o película. Este problema se puede atenuar retrasando el tiempo de inicio aún más, aunque hacer eso también requiere un búfer más grande. Los sitios Web comerciales que contienen transmisión continua de vídeo o audio utilizan reproductores que almacenan en el búfer por aproximadamente 10 seg antes de comenzar a reproducir.

Modelado de tráfico

En el ejemplo anterior, el origen envía los paquetes con un espaciado uniforme entre ellos, pero en otros casos, podrían emitirse de manera regular, lo cual puede causar congestión en la red. El envío no uniforme es común si el servidor está manejando muchos flujos al mismo tiempo, y también permite otras acciones, como avance rápido y rebobinado, autenticación de usuario,

etcétera. Además, el enfoque que utilizamos aquí (almacenamiento en el búfer) no siempre es posible, por ejemplo, en la videoconferencia. Sin embargo, si pudiera hacerse algo para hacer que el servidor (y los *hosts* en general) transmita a una tasa uniforme, la calidad del servicio mejoraría. A continuación examinaremos una técnica, el **modelado de tráfico,** que modera el tráfico en el servidor, en lugar de en el cliente.

El modelado de tráfico consiste en regular la *tasa* promedio (y las ráfagas) de la transmisión de los datos. En contraste, los protocolos de ventana corrediza que estudiamos anteriormente limitan la cantidad de datos en tránsito de una vez, no la tasa a la que se envían. Cuando se establece una conexión, el usuario y la subred (es decir, el cliente y la empresa portadora) acuerdan cierto patrón de tráfico (es decir, forma) para ese circuito. Algunas veces esto se llama **acuerdo de nivel de servicio.** En tanto el cliente cumpla con su parte del contrato y sólo envíe los paquetes acordados, la empresa portadora promete entregarlos de manera oportuna. El modelado de tráfico reduce la congestión y, por lo tanto, ayuda a la empresa portadora a cumplir con su promesa. Tales acuerdos no son tan importantes para las transferencias de archivos pero sí para los datos en tiempo real, como conexiones de vídeo y audio, lo cual tiene requerimientos rigurosos de calidad de servicio.

En efecto, con el modelado de tráfico, el cliente le dice a la empresa portadora: Mi patrón de transmisión se parecerá a esto, ¿puedes manejarlo? Si la empresa portadora acepta, surge la cuestión de cómo puede saber ésta si el cliente está cumpliendo con el acuerdo y cómo debe proceder si el cliente no lo está haciendo. La supervisión de un flujo de tráfico se conoce como **supervisión de tráfico** (*traffic policing*). Aceptar una forma de tráfico y supervisarlo más tarde es más fácil en las subredes de circuitos virtuales que en las de datagramas. Sin embargo, incluso en las subredes de datagramas se pueden aplicar las mismas ideas a las conexiones de la capa de transporte.

Algoritmo de cubeta con goteo

Imagínese una cubeta con un pequeño agujero en el fondo, como se ilustra en la figura 5-32(a). Sin importar la rapidez con que entra agua en la cubeta, el flujo de salida tiene una tasa constante, ρ, cuando hay agua en la cubeta, y una tasa de cero cuando la cubeta está vacía. Además, una vez que se llena la cubeta, cualquier agua adicional que entra se derrama por los costados y se pierde (es decir, no aparece en el flujo por debajo del agujero).

Puede aplicarse el mismo concepto a los paquetes, como se muestra en la figura 5-32(b). De manera conceptual, cada *host* está conectado a la red mediante una interfaz que contiene una cubeta con goteo, es decir, una cola interna infinita. Si llega un paquete cuando la cola está llena, éste se descarta. En otras palabras, si uno o más procesos del *host* tratan de enviar paquetes cuando la cola ya tiene la cantidad máxima de paquetes, dicho paquete se descarta sin más. Este arreglo puede incorporarse en la interfaz del hardware, o simularse a través del sistema operativo del *host*. El esquema fue propuesto inicialmente por Turner (1986), y se llama **algoritmo de cubeta con goteo.** De hecho, no es otra cosa que un sistema de encolamiento de un solo servidor con un tiempo de servicio constante.

El *host* puede poner en la red un paquete por pulso de reloj. Nuevamente, esto puede forzarse desde la tarjeta de interfaz o desde el sistema operativo. Este mecanismo convierte un flujo desi-

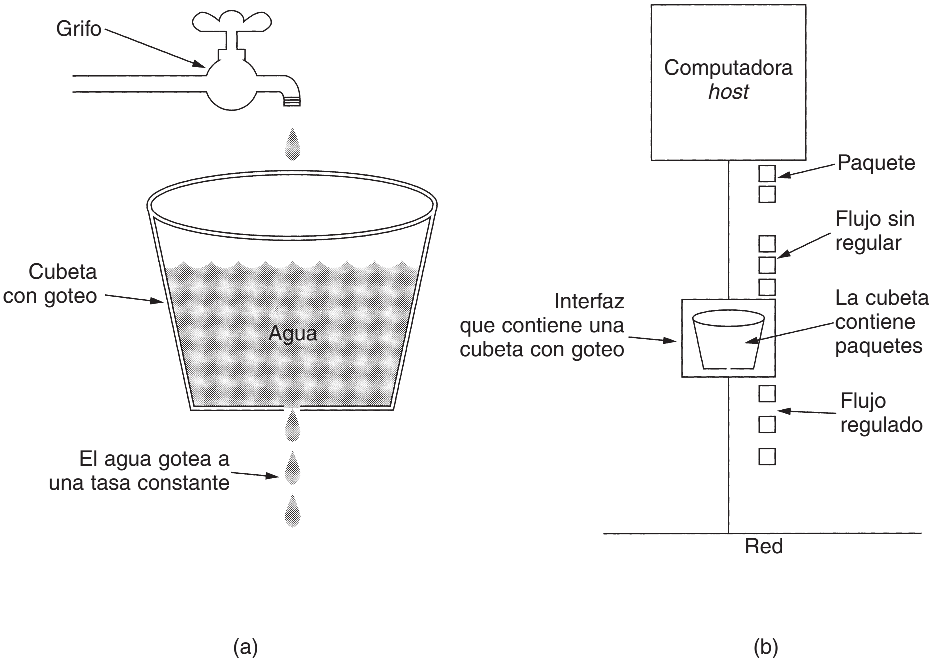


Figura 5-32. (a) Una cubeta con goteo, llena de agua. (b) Cubeta con goteo, llena de paquetes.

gual de paquetes de los procesos de usuario dentro del *host* en un flujo continuo de paquetes hacia la red, moderando las ráfagas y reduciendo en una buena medida las posibilidades de congestión.

Cuando todos los paquetes son del mismo tamaño (por ejemplo, celdas ATM), este algoritmo puede usarse como se describe. Sin embargo, cuando se utilizan paquetes de tamaño variable, con frecuencia es mejor permitir un número fijo de bytes por pulso, en lugar de un solo paquete. Por lo tanto, si la regla es de 1024 bytes por pulso, sólo pueden recibirse por pulso un paquete de 1024 bytes, dos paquetes de 512 bytes, cuatro paquetes de 256 bytes, etcétera. Si el conteo de bytes residuales es demasiado bajo, el siguiente paquete debe esperar hasta el siguiente pulso.

La implementación del algoritmo de cubeta con goteo es fácil. La cubeta con goteo consiste en una cola finita. Si cuando llega un paquete hay espacio en la cola, éste se agrega a ella; de otro modo, se descarta. En cada pulso de reloj se transmite un paquete (a menos que la cola esté vacía).

La cubeta con goteo que usa conteo de bits se implementa casi de la misma manera. En cada pulso un contador se inicializa en *n*. Si el primer paquete de la cola tiene menos bytes que el valor actual del contador, se transmite y se disminuye el contador en esa cantidad de bytes. Pueden enviarse paquetes adicionales en tanto el contador sea lo suficientemente grande. Cuando el contador está por debajo de la longitud del siguiente paquete de la cola, la transmisión se detiene hasta el siguiente pulso, en cuyo momento se restablece el conteo de bytes residuales y el flujo puede continuar.

Como ejemplo de cubeta con goteo, imagine que una computadora puede producir datos a razón de 25 millones de bytes/seg (200 Mbps) y que la red también opera a esta velocidad. Sin embargo, los enrutadores pueden manejar esta tasa de datos sólo durante intervalos cortos (básicamente, hasta que sus búferes se llenen). Durante intervalos grandes, dichos enrutadores funcionan mejor con tasas que no exceden 2 millones de bytes/seg. Ahora suponga que los datos llegan en ráfagas de un millón de bytes, con una ráfaga de 40 mseg cada segundo. Para reducir la tasa promedio a 2 MB/seg, podemos usar una cubeta con goteo de $\rho = 2$ MB/seg y una capacidad, C, de 1 MB. Esto significa que las ráfagas de hasta 1 MB pueden manejarse sin pérdidas de datos, ya que se distribuyen a través de 500 mseg, sin importar la velocidad a la que lleguen.

En la figura 5-33(a) vemos la entrada de la cubeta con goteo operando a 25 MB/seg durante 40 mseg. En la figura 5-33(b) vemos la salida drenándose a una velocidad uniforme de 2 MB/seg durante 500 mseg.

Algoritmo de cubeta con *tokens*

El algoritmo de cubeta con goteo impone un patrón de salida rígido a la tasa promedio, sin importar la cantidad de ráfagas que tenga el tráfico. En muchas aplicaciones es mejor permitir que la salida se acelere un poco cuando llegan ráfagas grandes, por lo que se necesita un algoritmo más flexible, de preferencia uno que nunca pierda datos. El **algoritmo de cubeta con *tokens*** es uno de tales algoritmos. En este algoritmo, la cubeta con goteo contiene *tokens*, generados por un reloj a razón de un *token* cada ΔT seg. En la figura 5-34(a) se muestra una cubeta que contiene tres *tokens* y cinco paquetes esperando a ser transmitidos. Para que se transmita un paquete, éste debe capturar y destruir un *token*. En la figura 5-34(b) vemos que han pasado tres de los cinco paquetes, pero los otros dos están atorados, esperando la generación de dos o más *tokens*.

El algoritmo de cubeta con *tokens* ofrece una forma diferente de modelado de tráfico que el algoritmo de cubeta con goteo. Este último no permite que los *hosts* inactivos acumulen permisos para enviar posteriormente ráfagas grandes. El algoritmo de cubeta con *tokens* sí permite el ahorro, hasta el tamaño máximo de la cubeta, n. Esta propiedad significa que pueden enviarse a la vez ráfagas de hasta n paquetes, permitiendo cierta irregularidad en el flujo de salida y dando una respuesta más rápida a las ráfagas de entrada repentinas.

Otra diferencia entre los dos algoritmos es que el algoritmo de cubeta con *tokens* descarta los *tokens* (es decir, la capacidad de transmisión) cuando se llena la cubeta, pero nunca descarta los paquetes. En contraste, el algoritmo de cubeta con goteo descarta los paquetes cuando se llena la cubeta.

Aquí también es posible una variación menor, en la que cada *token* representa el derecho de transmitir no un paquete, sino k bytes. Sólo puede transmitirse un paquete si hay suficientes *tokens* disponibles para cubrir su longitud en bytes. Los *tokens* fraccionarios se guardan para uso futuro.

Los algoritmos de cubeta con goteo y cubeta con *tokens* también pueden servir para regular el tráfico entre los enrutadores, así como para regular la salida de un *host*, como en nuestros ejemplos. Sin embargo, una diferencia clara es que una cubeta con *tokens* que regula a un *host* puede hacer que éste detenga el envío cuando las reglas dicen que debe hacerlo. Indicar a un enrutador que detenga la transmisión mientras sigue recibiendo entradas puede dar como resultado una pérdida de datos.

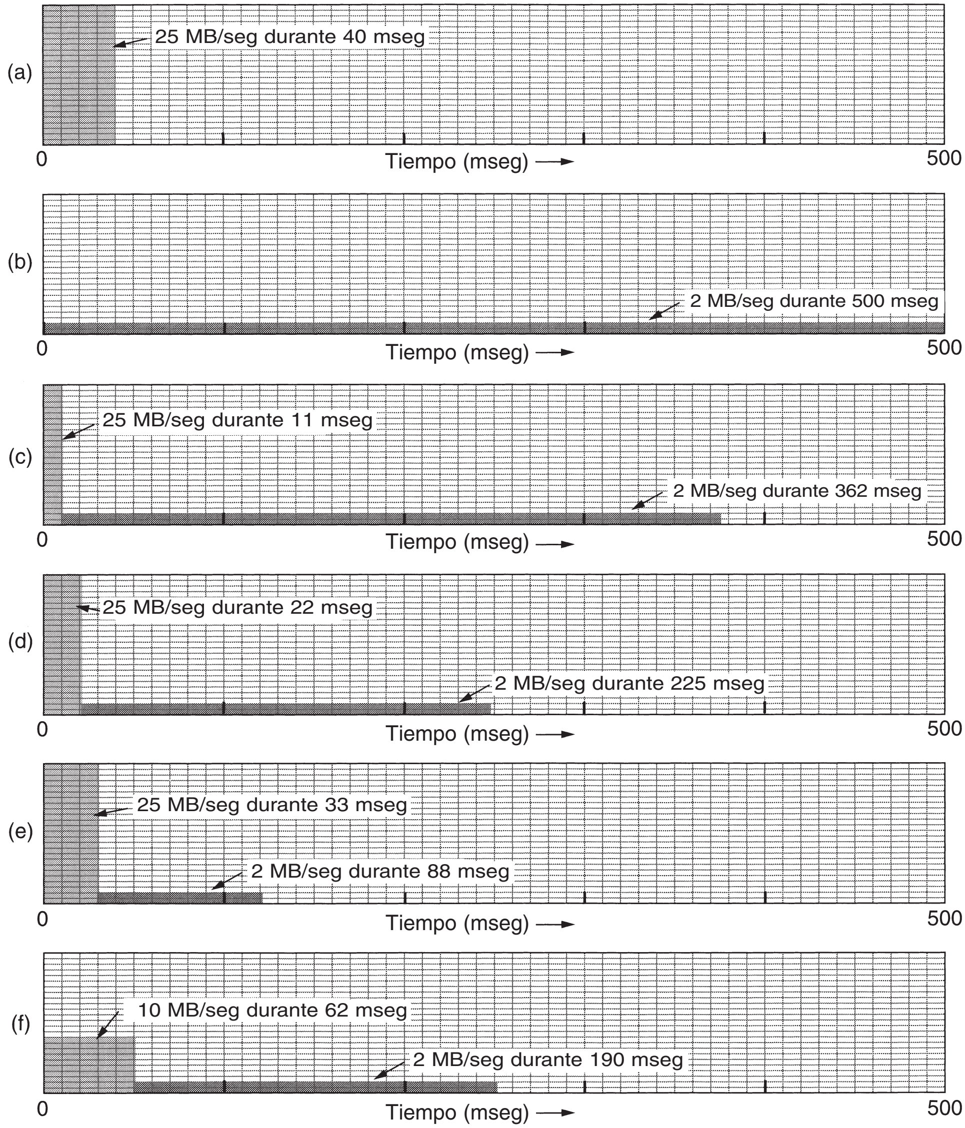


Figura 5-33. (a) Entrada a una cubeta con goteo. (b) Salida de una cubeta con goteo. (c)-(e) Salida de una cubeta con *tokens* con capacidades de 250 KB, 500 KB y 750 KB. (f) Salida de una cubeta con *tokens* de 500 KB que alimenta a una cubeta con goteo de 10 MB/seg.

La implementación del algoritmo básico de cubeta con *tokens* simplemente es sólo una variable que cuenta *tokens*. El contador se incrementa en uno cada ΔT y se decrementa en uno cada vez que se envía un paquete. Cuando el contador llega a cero, ya no pueden enviarse paquetes. En la variante de conteo de bits, el contador se incrementa en k bytes cada ΔT y se decrementa en la longitud de cada paquete enviado.

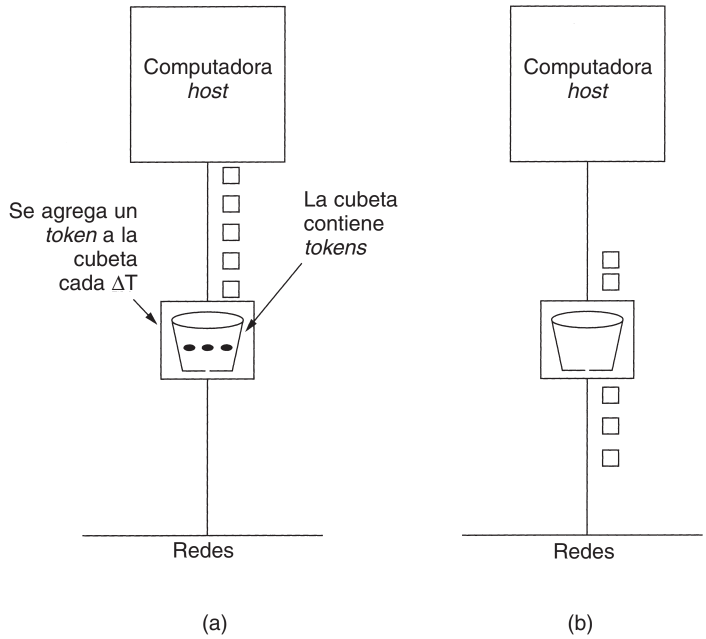


Figura 3-34. Algoritmo de cubeta con *tokens*. (a) Antes. (b) Después.

En esencia, lo que hace la cubeta con *tokens* es permitir ráfagas, pero limitadas a una longitud máxima regulada. Por ejemplo, vea la figura 5-33(c). Ahí tenemos una cubeta de *tokens* con 250 KB de capacidad. Los *tokens* llegan a una tasa que permite un flujo de salida de 2 MB/seg. Suponiendo que la cubeta con *tokens* está llena cuando llega la ráfaga de 1 MB, la cubeta puede drenarse a la velocidad máxima de 25 MB/seg durante unos 11 mseg. Entonces tiene que desacelerarse hasta 2 MB/seg hasta que se ha enviado toda la ráfaga de entrada.

El cálculo de la longitud de ráfaga con tasa máxima es un tanto complicado. No es sólo la división de 1 MB entre 25 MB/seg, ya que, mientras se está enviando la ráfaga, llegan más *tokens*. Si S seg es la longitud de la ráfaga, C bytes es la capacidad de la cubeta con *tokens*, ρ bytes/seg es la tasa de llegada de *tokens* y M bytes/seg es la tasa máxima de salida, podemos ver que una ráfaga de salida contiene un máximo de $C + \rho S$ bytes. También sabemos que la cantidad de bytes en una ráfaga a velocidad máxima con longitud de S segundos es MS. Por lo tanto, tenemos

$$C + \rho S = MS$$

Podemos resolver esta ecuación para obtener $S = C/(M - \rho)$. Para nuestros parámetros de $C = 250$ KB, $M = 25$ MB/seg y $\rho = 2$ MB/seg, tenemos un tiempo de ráfaga de aproximadamente 11 mseg. En la figura 5-33(d) y en la figura 5-33(e) se muestra la cubeta con *tokens* para capacidades de 500 y 750 KB, respectivamente.

Un problema potencial con el algoritmo de cubeta con *tokens* es que permite ráfagas largas, aunque puede regularse el intervalo máximo de ráfaga mediante una selección cuidadosa de ρ y *M*. Con frecuencia es deseable reducir la tasa pico, pero sin regresar al valor mínimo de la cubeta con goteo original.

Una manera de lograr tráfico más uniforme es poner una cubeta con goteo después de la cubeta con *tokens*. La tasa de la cubeta con goteo deberá ser mayor que la ρ de la cubeta con *tokens*, pero menor que la tasa máxima de la red. En la figura 5-33(f) se muestra la salida de una cubeta con *tokens* de 500 KB seguida de una cubeta con goteo de 10 MB/seg.

La supervisión de estos esquemas puede ser un tanto complicada. En esencia, la red tiene que simular el algoritmo y asegurarse de que no se envíen más paquetes o bytes de lo permitido. Sin embargo, estas herramientas proporcionan métodos para modelar el tráfico de la red de formas más manejables para ayudar a cumplir con los requerimientos de calidad del servicio.

Reservación de recursos

El hecho de tener la capacidad de regular la forma del tráfico ofrecido es un buen inicio para garantizar la calidad del servicio. Sin embargo, utilizar efectivamente esta información significa de manera implícita obligar a todos los paquetes de un flujo a que sigan la misma ruta. Su envío a través de enrutadores aleatorios dificulta garantizar algo. Como consecuencia, se debe configurar algo similar a un circuito virtual del origen al destino, y todos los paquetes que pertenecen al flujo deben seguir esta ruta.

Una vez que se tiene una ruta específica para una flujo, es posible reservar recursos a lo largo de esa ruta para asegurar que la capacidad necesaria esté disponible. Se pueden reservar tres tipos de recursos:

1. Ancho de banda.
2. Espacio de búfer.
3. Ciclos de CPU.

El primero, ancho de banda, es el más obvio. Si un flujo requiere 1 Mbps y la línea saliente tiene una capacidad de 2 Mbps, tratar de dirigir tres flujos a través de esa línea no va a funcionar. Por lo tanto, reservar ancho de banda significa no sobrecargar ninguna línea de salida.

Un segundo recurso que por lo general es escaso es el espacio en búfer. Cuando llega un paquete, por lo general el hardware mismo lo deposita en la tarjeta de interfaz de red. A continuación, el software enrutador tiene que copiarlo en un búfer en RAM y colocar en la cola ese búfer para transmitirlo en la línea saliente elegida. Si no hay búfer disponible, el paquete se tiene que descartar debido a que no hay lugar para colocarlo. Para una buena calidad de servicio, es posible reservar algunos búferes para un flujo específico de manera que éste no tenga que competir con otros flujos para obtener espacio en búfer. Siempre que ese flujo necesite un búfer, se le proporcionará uno mientras existan disponibles.

Por último, los ciclos de CPU también son un recurso escaso. Para procesar un paquete se necesita tiempo de CPU del enrutador, por lo que un enrutador sólo puede procesar cierta cantidad de paquetes por segundo. Para asegurar el procesamiento oportuno de cada paquete, es necesario verificar que la CPU no esté sobrecargada.

A primera vista podría parecer que si un enrutador tarda, digamos, 1 μseg, en procesar un paquete, entonces puede procesar 1 millón de paquetes/seg. Esta observación no es verdadera porque siempre habrá periodos inactivos debido a fluctuaciones estadísticas en la carga. Si la CPU necesita cada ciclo para poder realizar su trabajo, la pérdida incluso de algunos ciclos debido a periodos inactivos ocasionales crea un atraso del que nunca se podrá deshacer.

Sin embargo, incluso con una carga que esté ligeramente por debajo de la capacidad teórica, se pueden generar colas y pueden ocurrir retardos. Considere una situación en la que los paquetes llegan de manera aleatoria con una tasa promedio de llegada de λ paquetes/seg. El tiempo de CPU requerido por cada uno también es aleatorio, con una capacidad media de procesamiento de μ paquetes/seg. Bajo el supuesto de que las distribuciones de arribo y de servicio son distribuciones de Poisson, es posible probar, mediante la teoría de encolamiento, que el retardo promedio experimentado por un paquete, T, es

$$T = \frac{1}{\mu} \times \frac{1}{1 - \lambda/\mu} = \frac{1}{\mu} \times \frac{1}{1 - \rho}$$

donde $\rho = \lambda/\mu$ es el uso de CPU. El primer factor, $1/\mu$, sería el tiempo de servicio si no hubiera competencia. El segundo factor es la reducción de velocidad debido a la competencia con otros flujos. Por ejemplo, si $\lambda = 950{,}000$ paquetes/seg y $\mu = 1{,}000{,}000$ paquetes/seg, entonces $\rho = 0.95$ y el retardo promedio experimentado por cada paquete será de 20 μseg en lugar de 1 μseg. Este tiempo cuenta tanto para el tiempo de encolamiento como para el de servicio, como puede verse cuando la carga es muy baja ($\lambda/\mu \approx 0$). Si hay, digamos, 30 enrutadores a lo largo de la ruta del flujo, el retardo de encolamiento será de alrededor de 600 μseg.

Control de admisión

Ahora nos encontramos en el punto en que el tráfico entrante de algún flujo está bien modelado y puede seguir una sola ruta cuya capacidad puede reservarse por adelantado en los enrutadores a lo largo de la ruta. Cuando un flujo de este tipo se ofrece a un enrutador, éste tiene que decidir, con base en su capacidad y en cuántos compromisos tiene con otros flujos, si lo admite o lo rechaza.

La decisión de aceptar o rechazar un flujo no se trata simplemente de comparar el ancho de banda, los búferes o los ciclos requeridos por el flujo con la capacidad excedida del enrutador en esas tres dimensiones. Es más complicado que eso. Para empezar, aunque algunas aplicaciones podrían saber sobre sus requerimientos de ancho de banda, saben poco acerca de búferes o ciclos de CPU y, por esto, se necesita por lo menos una forma diferente de describir los flujos. Además, algunas aplicaciones son mucho más tolerantes con el incumplimiento ocasional de plazos que otras. Por último, algunas aplicaciones podrían estar dispuestas a negociar los parámetros del flujo y otras no. Por ejemplo, un visor de películas que por lo general se ejecuta a 30 cuadros/seg podría

estar dispuesto a ejecutar a 25 cuadros/seg si no hay suficiente ancho de banda para soportar 30 cuadros/seg. De manera similar, la cantidad de píxeles por cuadro y de ancho de banda de audio, entre otras propiedades, podría ser ajustable.

Debido a que muchas partes pueden estar involucradas en la negociación del flujo (el emisor, el receptor y todos los enrutadores a lo largo de la ruta), los flujos deben describirse de manera precisa en términos de parámetros específicos que se puedan negociar. Un conjunto de tales parámetros se conoce como **especificación de flujo.** Por lo general, el emisor (por ejemplo, el servidor de vídeo) produce una especificación de flujo que propone los parámetros que le gustaría utilizar. Conforme la especificación se propague por la ruta, cada enrutador la examina y modifica los parámetros conforme sea necesario. Las modificaciones sólo pueden reducir el flujo, no incrementarlo (por ejemplo, una tasa más baja de datos, no una más grande). Cuando llega al otro extremo, se pueden establecer los parámetros.

Como ejemplo de lo que puede estar en una especificación de flujo, considere el de la figura 5-35, que se basa en los RFCs 2210 y 2211. Tiene cinco parámetros, el primero de los cuales, la *Tasa de la cubeta con tokens*, es la cantidad de bytes por segundo que se colocan en la cubeta. Ésta es la tasa máxima que el emisor puede transmitir, promediada con respecto a un intervalo de tiempo largo.

Parámetro	Unidad
Tasa de la cubeta con *tokens*	Bytes/seg
Tamaño de la cubeta con *tokens*	Bytes
Tasa pico de datos	Bytes/seg
Tamaño mínimo de paquete	Bytes
Tamaño máximo de paquete	Bytes

Figura 5-35. Ejemplo de especificación de flujo.

El segundo parámetro es el tamaño de la cubeta en bytes. Por ejemplo, si la *Tasa de la cubeta con tokens* es de 1 Mbps y el *Tamaño de la cubeta con tokens* es de 500 KB, la cubeta se puede llenar de manera continua durante 4 seg antes de llenarse por completo (en caso de que no haya transmisiones). Cualesquier *tokens* enviados después de eso, se pierden.

El tercer parámetro, la *Tasa pico de datos*, es la tasa máxima de transmisiones tolerada, incluso durante intervalos de tiempo breves. El emisor nunca debe sobrepasar esta tasa.

Los últimos dos parámetros especifican los tamaños mínimo y máximo de paquetes, incluyendo los encabezados de la capa de red y de transporte (por ejemplo, TCP e IP). El tamaño mínimo es importante porque procesar cada paquete toma un tiempo fijo, aunque sea breve. Un enrutador debe estar preparado para manejar 10,000 paquetes/seg de 1 KB cada uno, pero no para manejar 100,000 paquetes/seg de 50 bytes cada uno, aunque esto representa una tasa de datos menor. El tamaño máximo de paquete es importante debido a las limitaciones internas de la red que no deben sobrepasarse. Por ejemplo, si parte de la ruta es a través de una Ethernet, el tamaño máximo del paquete se restringirá a no más de 1500 bytes, sin importar lo que el resto de la red puede manejar.

Una pregunta interesante es cómo convierte un enrutador una especificación de flujo en un conjunto de reservaciones de recursos específicos. Esa conversión es específica de la implementación y no está estandarizada. Suponga que un enrutador puede procesar 100,000 paquetes/seg. Si se le ofrece un flujo de 1 MB/seg con tamaños de paquete mínimo y máximo de 512 bytes, el enrutador puede calcular que puede transmitir 2048 paquetes/seg de ese flujo. En ese caso, debe reservar 2% de su CPU para ese flujo, o de preferencia más para evitar retardos largos de encolamiento. Si la política de un enrutador es nunca asignar más de 50% de su CPU (lo que implica un retardo con factor de dos, y ya está lleno el 49%, entonces ese flujo debe rechazarse). Se necesitan cálculos similares para los otros recursos.

Entre más rigurosa sea la especificación de flujo, más útil será para los enrutadores. Si una especificación de flujo especifica que necesita una *tasa de cubeta con tokens* de 5 MB/seg pero los paquetes pueden variar de 50 bytes a 1500 bytes, entonces la tasa de paquetes variará aproximadamente de 3500 a 105,000 paquetes/seg. El enrutador podría asustarse por este último número y rechazar el flujo, mientras que con un tamaño mínimo de paquete de 1000 bytes, el flujo de 5 MB/seg podría aceptarse.

Enrutamiento proporcional

La mayoría de los algoritmos de enrutamiento tratan de encontrar la mejor ruta para cada destino y envían a través de ella todo el tráfico a ese destino. Un método diferente que se ha propuesto para proporcionar una calidad de servicio más alta es dividir el tráfico para cada destino a través de diferentes rutas. Puesto que generalmente los enrutadores no tienen un panorama completo del tráfico de toda la red, la única forma factible de dividir el tráfico a través de múltiples rutas es utilizar la información disponible localmente. Un método simple es dividir el tráfico en fracciones iguales o en proporción a la capacidad de los enlaces salientes. Sin embargo, hay disponibles otros algoritmos más refinados (Nelakuditi y Zhang, 2002).

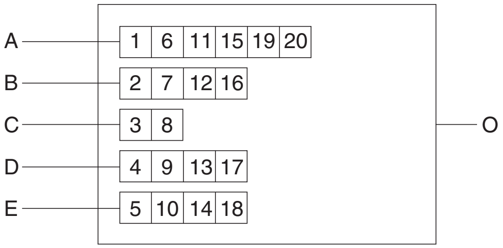
Calendarización de paquetes

Si un enrutador maneja múltiples flujos, existe el peligro de que un flujo acapare mucha de su capacidad y limite a los otros flujos. El procesamiento de paquetes en el orden de arribo significa que un emisor agresivo puede acaparar la mayor parte de la capacidad de los enrutadores por los que pasan sus paquetes, lo que reduce la calidad del servicio para otros. Para hacer fracasar esos intentos, se han diseñado varios algoritmos de programación de paquetes (Bhatti y Crowcroft, 2000).

Uno de los primeros fue el de **encolamiento justo** (*fair queueing*) (Nagle, 1987). La esencia del algoritmo es que los enrutadores tienen colas separadas para cada línea de salida, una por flujo. Cuando una línea se queda inactiva, el enrutador explora las diferentes colas de manera circular, y toma el primer paquete de la siguiente cola. De esta forma, con *n hosts* compitiendo por una línea de salida dada, cada *host* obtiene la oportunidad de enviar uno de *n* paquetes. El envío de más paquetes no mejorará esta fracción.

Aunque al principio este algoritmo tiene un problema: proporciona más ancho de banda a los *hosts* que utilizan paquetes más grandes que a los que utilizan paquetes más pequeños. Demers y cols. (1990) sugirieron una mejora en la que la exploración circular (*round robin*) se realiza de tal

manera que se simule una exploración circular byte por byte, en lugar de paquete por paquete. En efecto, explora las colas de manera repetida, byte por byte, hasta que encuentra el instante en el que finalizará cada paquete. A continuación, los paquetes se ordenan conforme a su tiempo de terminación y se envían en ese orden. En la figura 5-36 se ilustra este algoritmo.



(a)

Paquete	Tiempo de terminación
C	8
B	16
D	17
E	18
A	20

(b)

Figura 5-36. (a) Un enrutador con cinco paquetes encolados en la línea *O*. (b) Tiempos de terminación de los cinco paquetes.

En la figura 5-36(a) se muestran paquetes con una longitud de 2 hasta 6 bytes. En el pulso de reloj (virtual) 1, se envía el primer byte del paquete de la línea *A*. Después le toca el turno al primer byte del paquete de la línea *B*, y así sucesivamente. El primer paquete en terminar es *C*, después de ocho pulsos. El orden se muestra en la figura 5-36(b). Debido a que ya no hay más llegadas, los paquete se enviarán en el orden listado, de *C* a *A*.

Un problema con este algoritmo es que da la misma prioridad a todos los *hosts*. En muchas situaciones, es necesario dar a los servidores de vídeo más ancho de banda que a los servidores de archivos regulares, a fin de que puedan proporcionárseles dos o más bytes por pulso. Este algoritmo modificado se conoce como **encolamiento justo ponderado** (*weighted fair queueing*) y se utiliza ampliamente. Algunas veces el peso es igual a la cantidad de flujos provenientes de una máquina, de manera que el proceso obtiene un ancho de banda igual. Una implementación eficiente del algoritmo se analiza en (Shreedhar y Varghese, 1995). El reenvío real de paquetes a través de un enrutador o conmutador se está realizando cada vez más en el hardware (Elhanany y cols., 2001).

5.4.3 Servicios integrados

Entre 1995 y 1997, la IETF se esforzó mucho en diseñar una arquitectura para la multimedia de flujos continuos. Este trabajo resultó en cerca de dos docenas de RFCs, empezando con los RFCs 2205–2210. El nombre genérico para este trabajo es **algoritmos basados en flujo** o **servicios integrados.** Se diseñó tanto para aplicaciones de unidifusión como para multidifusión. Un ejemplo de la primera es un solo usuario difundiendo un clip de vídeo de un sitio de noticias. Un ejemplo del segundo es una colección de estaciones de televisión digital difundiendo sus programas como flujos de paquetes IP a muchos receptores de diferentes ubicaciones. A continuación nos concentraremos en la multidifusión, debido a que la transmisión por unidifusión es un caso especial de multidifusión.

En muchas aplicaciones de multidifusión, los grupos pueden cambiar su membresía de manera dinámica, por ejemplo, conforme las personas entran a una videoconferencia y se aburren y cambian a una telenovela o al canal del juego de croquet. Bajo estas condiciones, el método de hacer que los emisores reserven ancho de banda por adelantado no funciona bien, debido a que requeriría que cada emisor rastreara todas las entradas y salidas de su audiencia. Para un sistema diseñado para transmitir televisión con millones de suscriptores, ni siquiera funcionaría.

RSVP—Protocolo de reservación de recursos

El principal protocolo IETF para la arquitectura de servicios integrados es **RSVP.** Se describe en el RFC 2205 y en otros. Este protocolo se utiliza para marcar las reservas; para el envío de datos se utilizan otros protocolos. RSVP permite que varios emisores transmitan a múltiples grupos de receptores, permite que receptores individuales cambien de canal libremente, optimiza el uso de ancho de banda y elimina la congestión.

En su forma más sencilla, el protocolo usa enrutamiento de multidifusión con árboles de expansión, como se vio antes. A cada grupo se le asigna un grupo de direcciones. Para enviar a un grupo, un emisor pone la dirección del grupo en sus paquetes. El algoritmo estándar de multidifusión construye entonces un árbol de expansión que cubre a todos los miembros del grupo. El algoritmo de enrutamiento no es parte del RSVP. La única diferencia con la multidifusión normal es un poco de información extra multidifundida al grupo periódicamente para indicarle a los enrutadores a lo largo del árbol que mantengan ciertas estructuras de datos en sus memorias.

Como ejemplo, considere la red de la figura 5-37(a). Los *hosts* 1 y 2 son emisores multidifusión, y los *hosts* 3, 4 y 5 son receptores multidifusión. En este ejemplo, los emisores y los receptores son distintos pero, en general, los dos grupos pueden traslaparse. Los árboles de multidifusión de los *hosts* 1 y 2 se muestran en las figuras 5-37(b) y 5-37(c), respectivamente.

Para obtener mejor recepción y eliminar la congestión, cualquiera de los receptores de un grupo puede enviar un mensaje de reservación por el árbol al emisor. El mensaje se propaga usando el algoritmo de reenvío por ruta invertida estudiado antes. En cada salto, el enrutador nota la reservación y aparta el ancho de banda necesario; si no hay suficiente ancho de banda disponible, informa de una falla. En el momento que el mensaje llega de regreso al origen, se ha reservado el ancho de banda desde el emisor hasta el receptor que hace la solicitud de reservación a lo largo del árbol de expansión.

En la figura 5-38(a) se muestra un ejemplo de tales reservaciones. Aquí el *host* 3 ha solicitado un canal al *host* 1. Una vez establecido el canal, los paquetes pueden fluir de 1 a 3 sin congestiones. Ahora considere lo que sucede si el *host* 3 reserva a continuación un canal hacia otro emisor, el *host* 2, para que el usuario pueda ver dos programas de televisión a la vez. Se reserva una segunda ruta, como se muestra en la figura 5-38(b). Observe que se requieren dos canales individuales del *host* 3 al enrutador *E*, porque se están transmitiendo dos flujos independientes.

Por último, en la figura 5-38(c), el *host* 5 decide observar el programa transmitido por el *host* 1 y también hace una reservación. Primero se reserva ancho de banda dedicado hasta el enrutador *H*.

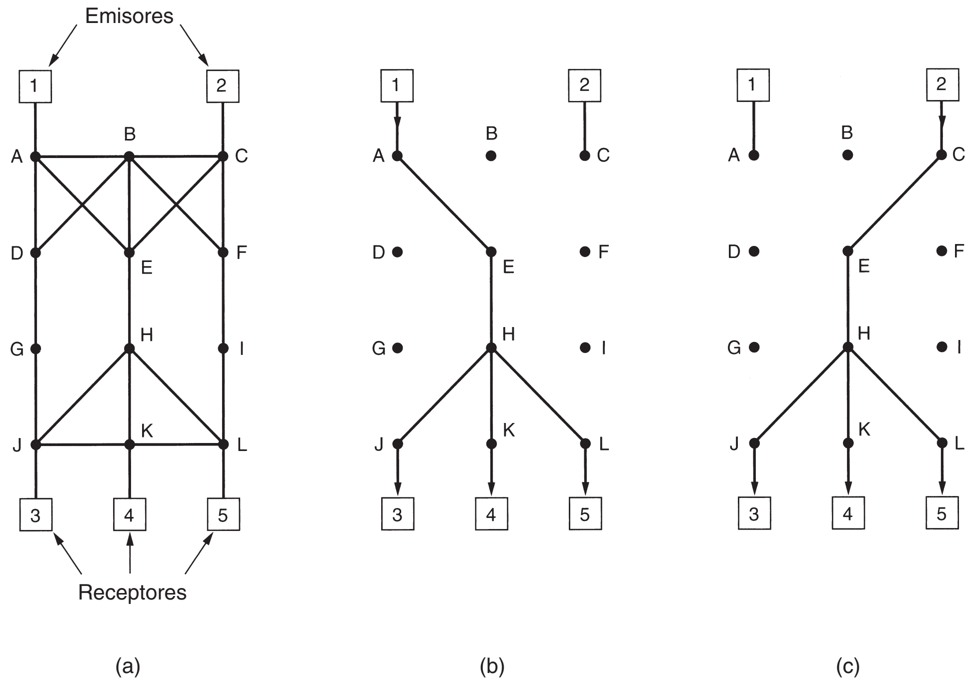


Figura 5-37. (a) Red. (b) Árbol de expansión de multidifusión para el *host* 1. (c) Árbol de expansión de multidifusión para el *host* 2.

Sin embargo, éste ve que ya tiene una alimentación del *host* 1, por lo que, si ya se ha reservado el ancho de banda necesario, no necesita reservar nuevamente. Observe que los *hosts* 3 y 5 podrían haber solicitado diferentes cantidades de ancho de banda (por ejemplo, el 3 tiene una televisión de blanco y negro, por lo que no quiere la información de color), así que la capacidad reservada debe ser lo bastante grande para satisfacer al receptor más voraz.

Al hacer una reservación, un receptor puede especificar (opcionalmente) uno o más orígenes de los que quiere recibir. También puede especificar si estas selecciones quedarán fijas durante toda la reservación, o si el receptor quiere mantener abierta la opción de cambiar los orígenes después. Los enrutadores usan esta información para optimizar la planeación del ancho de banda. En particular, sólo se establece que dos receptores van a compartir una ruta si ambos están de acuerdo en no cambiar los orígenes posteriormente.

La razón de esta estrategia en el caso totalmente dinámico es que el ancho de banda reservado está desacoplado de la selección del origen. Una vez que un receptor ha reservado ancho de banda, puede conmutarse a otro origen y conservar la parte de la ruta existente que es válida para el nuevo origen. Por ejemplo, si el *host* 2 está transmitiendo varios flujos de vídeo, el *host* 3 puede conmutarse entre ellos a voluntad sin cambiar su reservación: a los enrutadores no les importa el programa que está viendo el receptor.

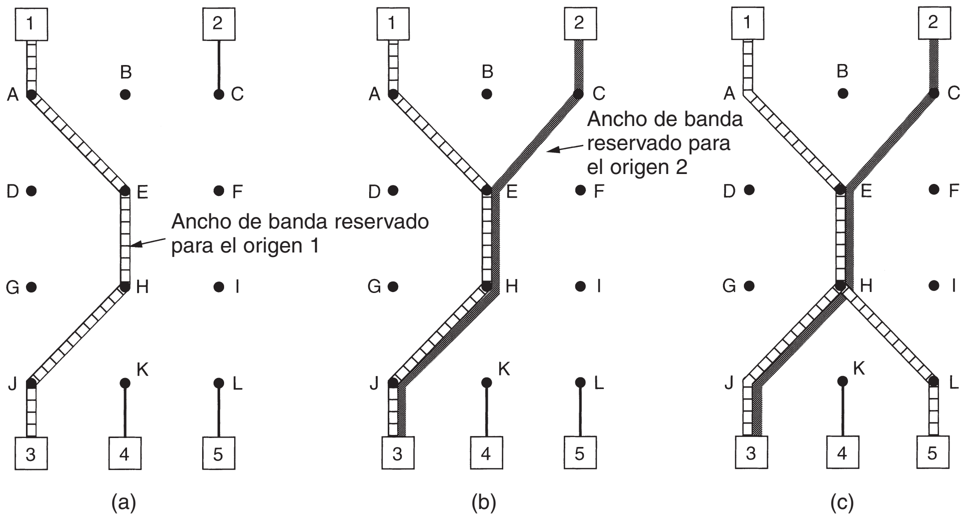


Figura 5-38. (a) El *host* 3 solicita un canal al *host* 1. (b) El *host* 3 solicita entonces un segundo canal al *host* 2. (c) El *host* 5 solicita un canal al *host* 1.

5.4.4 Servicios diferenciados

Los algoritmos basados en flujo tienen el potencial de ofrecer buena calidad de servicio a uno o más flujos debido a que reservan los recursos que son necesarios a lo largo de la ruta. Sin embargo, también tienen una desventaja. Requieren una configuración avanzada para establecer cada flujo, algo que no se escala bien cuando hay miles o millones de flujos. Además, mantienen estado por flujo interno en los enrutadores, haciéndolos vulnerables a las caídas de enrutadores. Por último, los cambios requeridos al código de enrutador son sustanciales e involucran intercambios complejos de enrutador a enrutador para establecer los flujos. Como consecuencia, existen pocas implementaciones de RSVP o algo parecido.

Por estas razones, la IETF también ha diseñado un método más simple para la calidad del servicio, uno que puede implementarse ampliamente de manera local en cada enrutador sin una configuración avanzada y sin que toda la ruta esté involucrada. Este método se conoce como calidad de servicio **basada en clase** (contraria a basada en flujo). La IETF ha estandarizado una arquitectura para él, llamada **servicios diferenciados,** que se describe en los RFCs 2474, 2475, entre otros. A continuación lo describiremos.

Un conjunto de enrutadores que forman un dominio administrativo (por ejemplo, un ISP o una compañía telefónica) puede ofrecer los servicios diferenciados (DS). La administración define un conjunto de clases de servicios con reglas de reenvío correspondientes. Si un cliente firma para un DS, los paquetes del cliente que entran en el dominio podrían contener un campo *Tipo de servicio*, con un mejor servicio proporcionado a algunas clases (por ejemplo, un servicio premium) que a otras. Al tráfico dentro de una clase se le podría requerir que se apegue a algún modelo específico, como a una cubeta con goteo con una tasa especificada de drenado. Un operador con intuición para los negocios

podría cargar una cantidad extra por cada paquete premium transportado o podría permitir hasta *N* paquetes premium por una mensualidad adicional fija. Observe que este esquema no requiere una configuración avanzada, ni reserva de recursos ni negociación extremo a extremo que consuma tiempo para cada flujo, como sucede con los servicios integrados. Esto hace de DS relativamente fácil de implementar.

El servicio basado en clase también ocurre en otras industrias. Por ejemplo, las compañías de envío de paquetes con frecuencia ofrecen servicio de tres días, de dos días, y servicio de un día para otro. Las aerolíneas ofrecen servicio de primera clase, de clase de negocios y de tercera clase. Los trenes que recorren largas distancias con frecuencia tienen múltiples clases de servicios. Incluso el metro de París tiene dos clases de servicios. Para los paquetes, las clases pueden diferir en términos de retardo, fluctuación y probabilidad de ser descartado en caso de congestión, entre otras posibilidades (pero probablemente sin tramas Ethernet más amplias).

Para hacer que la diferencia entre la calidad basada en el servicio y la basada en clase de servicio sea más clara, considere un ejemplo: la telefonía de Internet. Con un esquema basado en flujo, cada llamada telefónica obtiene sus propios recursos y garantías. Con un esquema basado en clase, todas las llamadas telefónicas obtienen los recursos reservados para la telefonía de clase. Estos recursos no pueden ser tomados por paquetes de la clase de transferencia de archivos u otras clases, pero ninguna llamada telefónica obtiene ningún recurso privado reservado sólo para ella.

Reenvío expedito o acelerado

Cada operador debe realizar la selección de clases de servicios, pero debido a que los paquetes con frecuencia se reenvían entre subredes ejecutadas por diferentes operadores, la IETF está trabajando para definir las clases de servicios independientes de la red. La clase más simple es el **reenvío expedito,** por lo tanto, iniciemos con ella. Se describe en el RFC 3246.

La idea detrás del reenvío expedito es muy simple. Dos clases de servicios están disponibles: regular y expedita. Se espera que la mayor parte del tráfico sea regular, pero una pequeña fracción de los paquetes son expeditos. Los paquetes expeditos deben tener la capacidad de transitar la subred como si no hubieran otros paquetes. En la figura 5-39 se muestra una representación simbólica de este sistema de “dos tubos”. Observe que todavía hay una línea física. Los dos conductos lógicos que se muestran en la figura representan una forma de reservar ancho de banda, no una segunda línea física.

Una forma de implementar esta estrategia es programar los enrutadores para que tengan dos colas de salida por cada línea de salida, una para los paquetes expeditos y una para los regulares. Cuando llega un paquete, se coloca en la cola de manera acorde. La programación de paquetes debe utilizar algo parecido al encolamiento justo ponderado. Por ejemplo, si 10% del tráfico es expedito y 90% es regular, 20% del ancho de banda podría dedicarse al tráfico expedito y el resto al tráfico regular. Al hacer esto se daría al tráfico expedito dos veces más ancho de banda del que necesita a fin de que dicho tráfico tenga un retardo bajo. Esta asignación se puede alcanzar transmitiendo un paquete expedito por cada cuatro paquetes regulares (suponiendo que el tamaño de la distribución para ambas clases es similar). De esta forma, se espera que los paquetes expeditos vean una red descargada, incluso cuando hay, de hecho, una carga pesada.

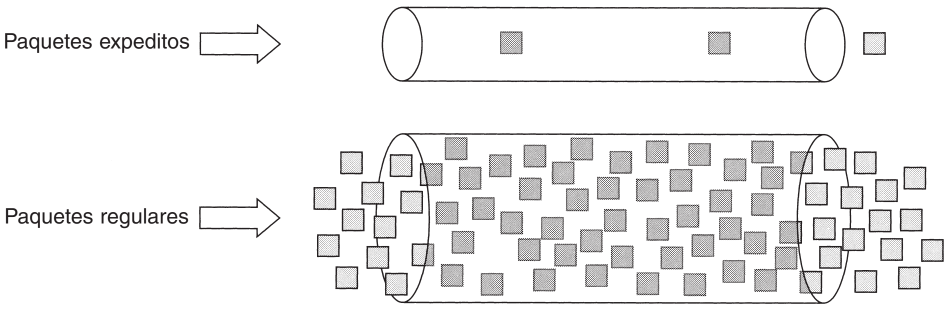


Figura 5-39. Los paquetes expeditos viajan por una red libre de tráfico.

Reenvío asegurado

Un esquema un poco más elaborado para el manejo de las clases de servicios se conoce como **reenvío asegurado.** Se describe en el RFC 2597. Especifica que deberán haber cuatro clases de prioridades, y cada una tendrá sus propios recursos. Además, define tres probabilidades de descarte para paquetes que están en congestión: baja, media y alta. En conjunto, estos dos factores definen 12 clases de servicios.

La figura 5-40 muestra una forma en que los paquetes pueden ser procesados bajo reenvío asegurado. El paso 1 es clasificar los paquetes en una de cuatro clases de prioridades. Este paso podría realizarse en el *host* emisor (como se muestra en la figura) o en el enrutador de ingreso. La ventaja de realizar la clasificación en el *host* emisor es que hay más información disponible acerca de cuáles paquetes pertenecen a qué flujos.

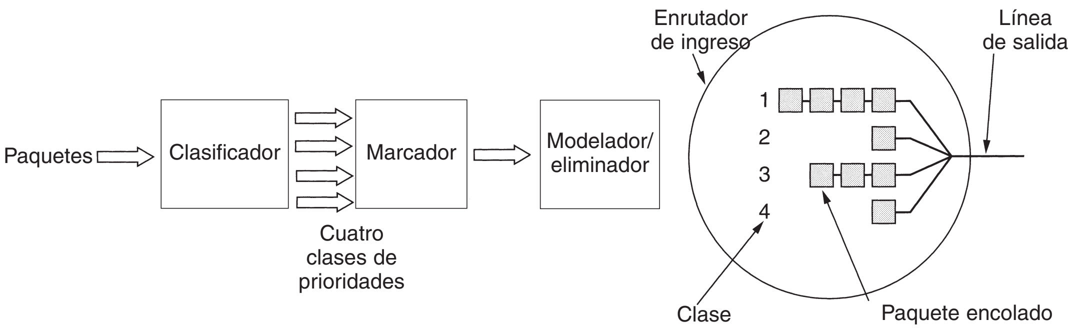


Figura 5-40. Una posible implementación del flujo de datos para el reenvío asegurado.

El paso 2 es marcar los paquetes de acuerdo con su clase. Para este propósito se necesita un campo de encabezado. Por fortuna, en el encabezado IP está disponible un campo *Tipo de servicio* de 8 bits, como veremos un poco más adelante. El RFC 2597 especifica que seis de estos bits se van a utilizar para la clase de servicio, dejando espacio de codificación para clases de servicio históricas y para futuras.

El paso 3 es pasar los paquetes a través de un filtro modelador/eliminador que podría retardar o descartar algunos de ellos para dar una forma aceptable a los cuatro flujos, por ejemplo, mediante cubetas con goteo o con *tokens*. Si hay muchos paquetes, algunos de ellos podrían descartarse aquí, mediante una categoría de eliminación. También son posibles esquemas elaborados que involucren la medición o la retroalimentación.

En este ejemplo, estos tres pasos se realizan en el *host* emisor, por lo que el flujo de salida ahora se introduce en el enrutador de ingreso. Vale la pena mencionar que estos pasos pueden ser realizados por software especial de conectividad de redes o incluso por el sistema operativo, a fin de no tener que cambiar las aplicaciones existentes.

5.4.5 Conmutación de etiquetas y MPLS

Mientras la IETF estaba desarrollando servicios integrados y diferenciados, varios fabricantes de enrutadores estaban desarrollando mejores métodos de reenvío. Este trabajo se enfocó en agregar una etiqueta en frente de cada paquete y realizar el enrutamiento con base en ella y no con base en la dirección de destino. Hacer que la etiqueta sea un índice de una tabla provoca que encontrar la línea correcta de salida sea una simple cuestión de buscar en una tabla. Al utilizar esta técnica, el enrutamiento puede llevarse a cabo de manera muy rápida y cualesquier recursos necesarios pueden reservarse a lo largo de la ruta.

Por supuesto, etiquetar los flujos de esta manera se acerca peligrosamente a los circuitos virtuales. X.25, ATM, frame relay, y otras redes con una subred de circuitos virtuales colocan una etiqueta (es decir, un identificador de circuitos virtuales) en cada paquete, la buscan en una tabla y enrutan con base en la entrada de la tabla. A pesar del hecho de que muchas personas en la comunidad de Internet tienen una aversión intensa por las redes orientadas a la conexión, la idea parece surgir nuevamente, pero esta vez para proporcionar un enrutamiento rápido y calidad de servicio. Sin embargo, hay diferencias esenciales entre la forma en que Internet maneja la construcción de la ruta y la forma en que lo hacen las redes orientadas a la conexión, por lo que esta técnica no utiliza la conmutación de circuitos tradicional.

Esta "nueva" idea de conmutación ha pasado por varios nombres (propietarios), entre ellos **conmutación de etiquetas.** En algún momento, la IETF comenzó a estandarizar la idea bajo el nombre **MPLS** (**conmutación de etiquetas multiprotocolo**)**.** De aquí en adelante lo llamaremos MPLS. Se describe en el RFC 3031, entre muchos otros.

Además, algunas personas hacen una distinción entre *enrutamiento* y *conmutación*. El enrutamiento es el proceso de buscar una dirección de destino en una tabla para saber a dónde enviar los paquetes hacia ese destino. En contraste, la conmutación utiliza una etiqueta que se toma de un paquete como un índice en una tabla de reenvío. Sin embargo, estas definiciones están lejos de ser universales.

El primer problema es en dónde colocar la etiqueta. Debido a que los paquetes IP no fueron diseñados para circuitos virtuales, en el encabezado IP no hay ningún campo disponible para los números de tales circuitos. Por esta razón, se tuvo que agregar un nuevo encabezado MPLS enfrente del encabezado IP. En una línea de enrutador a enrutador que utiliza PPP como protocolo

de tramas, el formato de trama, incluyendo los encabezados PPP, MPLS, IP y TCP, es como se muestra en la figura 5-41. De cierta forma, MPLS es, por lo tanto, la capa 2.5.

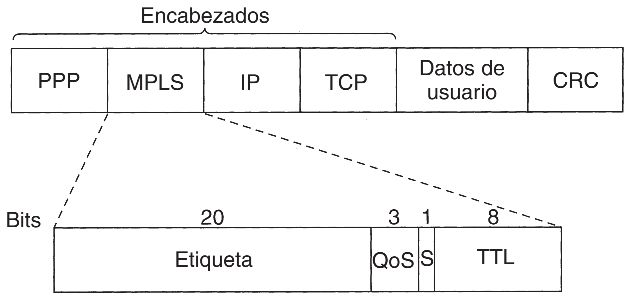


Figura 5-41. Transmisión de un segmento TCP que utiliza IP, MPLS y PPP.

El encabezado MPLS genérico tiene cuatro campos, el más importante de los cuales es el de *Etiqueta*, el cual contiene el índice. El campo *QoS (bits experimentales)* indica la clase de servicio. El campo *S* se relaciona con colocar en una pila múltiples etiquetas en redes jerárquicas (que se analizan más adelante). Si tiene el valor de 1 indica que es la última etiqueta añadida al paquete IP, si es un 0 indica que hay más etiquetas añadidas al paquete. El campo evita el ciclo infinito en caso de que haya inestabilidad en el enrutamiento, ya que se decrementa en cada enrutador y al llegar al valor de 0, el paquete es descartado.

Debido a que los encabezados MPLS no son parte del paquete de la capa de red o de la trama del enlace de datos, MPLS es en gran medida independiente de ambas capas. Entre otras cosas, esta propiedad significa que es posible construir conmutadores MPLS que pueden reenviar tanto paquetes IP como celdas ATM, dependiendo de lo que aparezca. De esta característica proviene la parte “multiprotocolo” del nombre MPLS.

Cuando un paquete mejorado con MPLS (o celda) llega a un enrutador con capacidad MPLS, la etiqueta se utiliza como un índice en una tabla para determinar la línea de salida y la nueva etiqueta a utilizar. Esta conmutación de etiquetas se utiliza en todas las subredes de circuitos virtuales, debido a que las etiquetas sólo tienen importancia local y dos enrutadores diferentes pueden asignar la misma etiqueta a paquetes hacia diferentes destinos, es decir, la etiqueta es reasignada a la salida de cada enrutador, por lo que no se mantiene la misma etiqueta en toda la ruta. En la figura 5-3 vimos en acción este mecanismo. MPLS utiliza la misma técnica.

Una diferencia con respecto a los circuitos virtuales tradicionales es el nivel de agregación. Ciertamente es posible que cada flujo tenga su propio conjunto de etiquetas a través de la subred. Sin embargo, es más común que los enrutadores agrupen múltiples flujos que terminan en un enrutador o una LAN particulares y utilizan una sola etiqueta de ellos. Se dice que los flujos que están agrupados en una sola etiqueta pertenecen a la misma **FEC** (**clase de equivalencia de reenvío**). Esta clase cubre no sólo a dónde van los paquetes, sino también su clase de servicio (en el sentido de los servicios diferenciados), debido a que todos sus paquetes se tratan de la misma forma para propósitos de reenvío.

Con el enrutamiento de circuitos virtuales tradicional no es posible agrupar en el mismo identificador de circuitos virtuales varias rutas diferentes con diferentes puntos finales, debido a que podría no haber forma de distinguirlas en el destino final. Con MPLS, los paquetes aún contienen su dirección de destino final, además de la etiqueta, a fin de que al final de la red de MPLS pueda eliminarse la etiqueta y que el reenvío pueda continuar de la forma normal, utilizando la dirección de destino de la capa de red.

Una diferencia principal entre MPLS y los diseños de circuitos virtuales convencionales es la forma en que está construida la tabla de reenvío. En las redes de circuitos virtuales tradicionales, cuando un usuario desea establecer una conexión, se inicia un paquete de configuración en la red para crear la ruta y crear las entradas de la tabla de reenvío. MPLS no funciona de esa forma porque no hay fase de configuración para cada conexión (pues eso podría romper con la operación de mucho software existente en Internet).

En su lugar, hay dos formas de crear las entradas de la tabla de reenvío. En el método **orientado a datos,** cuando un paquete llega, el primer enrutador que encuentra contacta al siguiente enrutador en el sentido descendente del flujo a donde tiene que ir el paquete y le pide que genere una etiqueta para el flujo. Este método se aplica de manera recursiva. En efecto, ésta es una creación de circuitos virtuales por petición.

Los protocolos que hacen esta propagación son muy cuidadosos para evitar los ciclos cerrados (loops). Por lo general, utilizan una técnica llamada **subprocesos con color** (*colored threads*). La propagación en reversa de una FEC se puede comparar con extraer un subproceso de un color único en la subred. Si un enrutador ve un color que ya tiene, sabe que hay un ciclo y toma una medida para solucionarlo. El método dirigido por datos se utiliza principalmente en redes en las que el transporte subyacente es ATM (como sucede en la mayor parte del sistema telefónico).

La otra forma, que se utiliza en las redes que no se basan en ATM, es el método **dirigido por control.** Tiene algunas variantes. Una de ellas funciona de la siguiente manera. Cuando se inicia un enrutador, verifica para cuáles rutas es el último salto (por ejemplo, qué *hosts* están en su LAN). Después crea una o más FECs para ellas, asigna una etiqueta para cada una y pasa las etiquetas a sus vecinos. Éstos, a su vez, introducen las etiquetas en sus tablas de reenvío y envían nuevas etiquetas a sus vecinos, hasta que todos los enrutadores han adquirido la ruta. También es posible reservar recursos conforme la ruta está construida para garantizar una calidad de servicio apropiada.

MPLS puede operar a múltiples niveles al mismo tiempo. En el nivel más alto, cada empresa portadora puede considerarse como un tipo de metaenrutador, con una ruta a través de los metaenrutadores del origen al destino. Esta ruta puede utilizar MPLS. Sin embargo, MPLS también puede utilizarse dentro de la red de cada empresa portadora, lo que resulta en un segundo nivel de etiquetado. De hecho, un paquete puede llevar consigo una pila entera de etiquetas. El bit *S* de la figura 5-41 permite que un enrutador elimine una etiqueta para saber si quedaron etiquetas adicionales. Se establece a 1 para la etiqueta inferior y 0 para las otras etiquetas. En la práctica, esta característica se utiliza principalmente para implementar redes privadas virtuales y túneles recursivos.

Aunque las ideas básicas detrás de MPLS son directas, los detalles son extremadamente complicados, y tienen muchas variaciones y optimizaciones, por lo que ya no trataremos más ese tema. Para mayor información, vea (Davie y Rekhter, 2000; Lin y cols., 2002; Pepelnjak y Guichard, 2001, y Wang, 2001).

5.5 INTERCONECTIVIDAD

Hasta ahora hemos supuesto de manera implícita que hay una sola red homogénea y que cada máquina usa el mismo protocolo en cada capa. Por desgracia, este supuesto es demasiado optimista. Existen muchas redes diferentes, entre ellas LANs, MANs y WANs. En cada capa hay numerosos protocolos de uso muy difundido. En las siguientes secciones estudiaremos con cuidado los problemas que surgen cuando dos o más redes se juntan, formando una **interred.**

Existe una controversia considerable sobre si la abundancia actual de tipos de red es una condición temporal que desaparecerá tan pronto como todo mundo se dé cuenta de lo maravillosa que es [indique aquí su red favorita], o si es una característica inevitable pero permanente del mundo, que está aquí para quedarse. Tener diferentes redes invariablemente implica tener diferentes protocolos.

Creemos que siempre habrá una variedad de redes (y, por lo tanto, de protocolos) diferentes, por las siguientes razones. Antes que nada, la base instalada de redes diferentes es grande. Casi todas las instalaciones UNIX ejecutan TCP/IP. Muchos negocios grandes aún tienen *mainframes* que ejecutan SNA de IBM. Una cantidad considerable de compañías telefónicas operan redes ATM. Algunas LANs de computadoras personales aún usan Novell NCP/IPX o AppleTalk. Por último, las redes inalámbricas constituyen un área nueva y en desarrollo con una variedad de protocolos. Esta tendencia continuará por años, debido a problemas de herencia, tecnología nueva y al hecho de que no todos los fabricantes se interesan en que sus clientes puedan migrar fácilmente al sistema de otro fabricante.

Segundo, a medida que las computadoras y las redes se vuelven más baratas, el lugar de la toma de decisiones se desplaza hacia abajo. Muchas compañías tienen políticas en el sentido de que las compras de más de un millón de dólares tienen que ser aprobadas por la gerencia general, las compras de más de 100,000 dólares tienen que ser aprobadas por la gerencia media, pero las compras por debajo de 100,000 dólares pueden ser hechas por los jefes de los departamentos sin aprobación de los superiores. Esto puede dar como resultado fácilmente a que el departamento de ingeniería instale estaciones de trabajo de UNIX que ejecuten TCP/IP y a que el departamento de marketing instale Macs con AppleTalk.

Tercero, las distintas redes (por ejemplo, ATM e inalámbricas) tienen tecnologías radicalmente diferentes, por lo que no debe sorprendernos que, a medida que haya avances nuevos en hardware, también se cree software nuevo adaptado al nuevo hardware. Por ejemplo, la casa típica ahora es como la oficina típica de hace 10 años: está llena de computadoras que no se hablan entre ellas. En el futuro, podría ser común que el teléfono, la televisión y otros aparatos estuvieran en red, para controlarlos de manera remota. Esta nueva tecnología sin duda generará nuevos protocolos y redes.

Como ejemplo de cómo se pueden conectar redes diferentes, considere la figura 5-42. Ahí vemos una red corporativa con múltiples ubicaciones enlazadas por una red ATM de área amplia. En una de las ubicaciones, se utiliza una red dorsal óptica FDDI para conectar una Ethernet, una LAN inalámbrica 802.11 y la red de *mainframe* SNA del centro de datos corporativo.

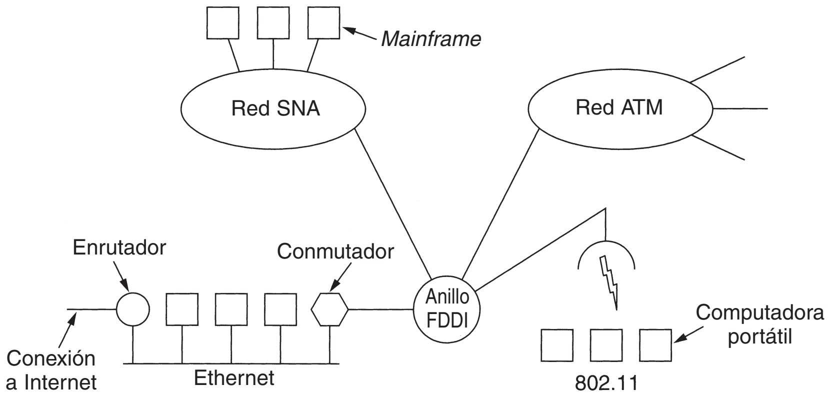


Figura 5-42. Una colección de redes interconectadas.

El propósito de interconectar todas estas redes es permitir que los usuarios de cualquiera de ellas se comuniquen con los usuarios de las demás, así como permitir que los usuarios de cualquiera de ellas accedan los datos de las demás. Lograr este objetivo significa enviar paquetes de una red a otra. Debido a que las redes, por lo general, difieren de formas considerables, obtener paquetes de una red a otra no siempre es tan fácil, como veremos a continuación.

5.5.1 Cómo difieren las redes

Las redes pueden diferir de muchas maneras. Algunas de las diferencias, como técnicas de modulación o formatos de tramas diferentes, se encuentran en las capas de enlace de datos y en la física. No trataremos esas diferencias aquí. En su lugar, en la figura 5-43 listamos algunas diferencias que pueden ocurrir en la capa de red. La conciliación de estas diferencias es lo que hace más difícil la interconexión de redes que la operación con una sola red.

Cuando los paquetes enviados por un origen en una red deben transitar a través de una o más redes foráneas antes de llegar a la red de destino (que también puede ser diferente de la red de origen), pueden ocurrir muchos problemas en las interfaces entre las redes. Para comenzar, cuando los paquetes de una red orientada a la conexión deben transitar a una red sin conexiones, deben reordenarse, algo que el emisor no espera y que el receptor no está preparado para manejar. Con frecuencia se necesitarán conversiones de protocolo, que pueden ser difíciles si la funcionalidad requerida no puede expresarse. También se necesitarán conversiones de direcciones, lo que podría requerir algún tipo de sistema de directorio. El paso de paquetes multidifusión a través de una red que no reconoce la multidifusión requiere la generación de paquetes individuales para cada destino.

Los diferentes tamaños máximos de paquete usados por las diferentes redes son un dolor de cabeza importante. ¿Cómo se pasa un paquete de 8000 bytes a través de una red cuyo tamaño máximo de paquete es de 1500 bytes? Es importante la diferencia en la calidad de servicio cuando

Aspecto	Algunas posibilidades
Servicio ofrecido	Sin conexiones, orientado a conexiones
Protocolos	IP, IPX, SNA, ATM, MPLS, AppleTalk, etc.
Direccionamiento	Plano (802) o jerárquico (IP)
Multidifusión	Presente o ausente (también difusión)
Tamaño de paquete	Cada red tiene su propio máximo
Calidad del servicio	Puede estar presente o ausente; muchos tipos diferentes
Manejo de errores	Entrega confiable, ordenada y desordenada
Control de flujo	Ventana corrediza, control de tasa, otros o ninguno
Control de congestión	Cubeta con goteo, paquetes reguladores, etc.
Seguridad	Reglas de confidencialidad, encriptación, etc.
Parámetros	Diferentes terminaciones de temporizador, especificaciones de flujo, etc.
Contabilidad	Por tiempo de conexión, por paquete, por byte, o sin ella

Figura 5-43. Algunas de las muchas maneras en que pueden diferir las redes.

un paquete que tiene restricciones de tiempo real pasa a través de una red que no ofrece garantías de tiempo real.

El control de errores, de flujo y de congestión suele ser diferente entre las diferentes redes. Si tanto el origen como el destino esperan la entrega de paquetes en secuencia y sin errores, pero una red intermedia simplemente descarta paquetes cuando huele congestión en el horizonte, o los paquetes vagan sin sentido durante un rato y emergen repentinamente para ser entregados, muchas aplicaciones fallarán. Existen diferentes mecanismos de seguridad, ajustes de parámetros y reglas de contabilidad, e incluso leyes de confidencialidad internacionales, que pueden causar problemas.

5.5.2 Conexión de redes

Las redes pueden interconectarse mediante diversos dispositivos, como vimos en el capítulo 4. Revisemos brevemente ese material. En la capa física, las redes se pueden conectar mediante repetidores o concentradores, los cuales mueven los bits de una red a otra idéntica. Éstos son en su mayoría dispositivos analógicos y no comprenden nada sobre protocolos digitales (simplemente regeneran señales).

En la capa de enlace de datos encontramos puentes y conmutadores. Pueden aceptar tramas, examinar las direcciones MAC y reenviar las tramas a una red diferente mientras realizan una traducción menor de protocolos en el proceso, por ejemplo, de Ethernet a FDDI o a 802.11.

En la capa de red hay enrutadores que pueden conectar dos redes. Si éstas tienen capas de red diferentes, el enrutador puede tener la capacidad de traducir entre los formatos de paquetes, aunque la traducción de paquetes ahora es cada vez menos común. Un enrutador que puede manejar múltiples protocolos se conoce como **enrutador multiprotocolo.**

En la capa de transporte se encuentran puertas de enlace de transporte, que pueden interactuar entre dos conexiones de transporte. Por ejemplo, una puerta de enlace de transporte podría permitir que los paquetes fluyeran entre una red TCP y una SNA, las cuales tienen protocolos de transporte diferentes, fijando esencialmente una conexión TCP con una SNA.

Por último, en la capa de aplicación, las puertas de enlace de aplicación traducen semánticas de mensaje. Como ejemplo, las puertas de enlace entre el correo electrónico de Internet (RFC 822) y el correo electrónico X.400 deben analizar los mensajes de correo electrónico y cambiar varios campos de encabezado.

En este capítulo nos enfocaremos en la interconectividad en la capa de red. Para ver cómo difiere esto de la conmutación en la capa de enlace de datos, examine la figura 5-44. En la figura 5-44(a), la máquina de origen, *S*, desea enviar un paquete a la máquina de destino, *D*. Estas máquinas se encuentran en Ethernets diferentes, conectadas mediante un conmutador. *S* encapsula el paquete en una trama y lo envía a su destino. La trama llega al conmutador, el cual ve la dirección MAC de dicha trama y determina que ésta tiene que ir a la LAN 2. El conmutador elimina la trama de la LAN 1 y la coloca en la LAN 2.

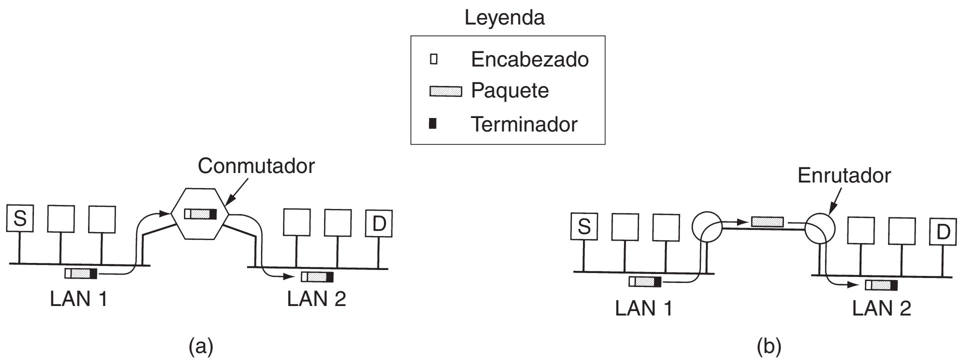


Figura 5-44. (a) Dos Ethernets conectadas mediante un conmutador. (b) Dos Ethernets conectadas mediante enrutadores.

Ahora consideremos la misma situación pero con las dos Ethernets conectadas mediante un par de enrutadores en lugar de un conmutador. Los enrutadores se conectan mediante una línea punto a punto, posiblemente una línea rentada de miles de kilómetros de longitud. Ahora el enrutador recoge la trama y el paquete se elimina del campo de datos de dicha trama. El enrutador examina la dirección del paquete (por ejemplo, una dirección IP) y la busca en su tabla de enrutamiento. Con base en esta dirección, decide enviar el paquete al enrutador remoto, encapsulado en un tipo diferente de trama, dependiendo del protocolo de línea. En el otro extremo el paquete se coloca en el campo de datos de una trama Ethernet y se deposita en la LAN 2.

Lo anterior es la diferencia esencial entre el caso de conmutación (o puenteo) y el caso enrutado. Con un conmutador (o puente), toda la trama se transporta con base en su dirección MAC. Con un enrutador, el paquete se extrae de la trama y la dirección del paquete se utiliza para decidir

a dónde enviarlo. Los conmutadores no tienen que entender el protocolo de capa de red que se está utilizando para conmutar los paquetes. Los enrutadores sí tienen que hacerlo.

5.5.3 Circuitos virtuales concatenados

Dos estilos posibles de interconectividad: la concatenación orientada a la conexión de subredes de circuitos virtuales, y los datagramas estilo Internet. A continuación los examinaremos por separado, pero primero una advertencia. En el pasado, la mayoría de las redes (públicas) eran orientadas a la conexión (frame relay, SNA, 802.16 y ATM aún lo son). Posteriormente, con la aceptación rápida de Internet, los datagramas se pusieron de moda. Sin embargo, sería un error pensar que los datagramas son para siempre. En este negocio, lo único que es para siempre es el cambio. Con la importancia creciente de las redes de multimedia, es probable que la orientación a la conexión regrese de una forma o de otra, puesto que es más fácil garantizar la calidad de servicio con conexiones que sin ellas. Por lo tanto, dedicaremos algún tiempo para estudiar las redes orientadas a la conexión.

En el modelo de circuitos virtuales concatenados, que se muestra en la figura 5-45, se establece una conexión con un *host* de una red distante de un modo parecido a la manera en que se establecen normalmente las conexiones. La subred ve que el destino es remoto y construye un circuito virtual al enrutador más cercano a la red de destino; luego construye un circuito virtual de ese enrutador a una **puerta de enlace** externa (enrutador multiprotocolo). Ésta registra la existencia del circuito virtual en sus tablas y procede a construir otro circuito virtual a un enrutador de la siguiente subred. Este proceso continúa hasta llegar al *host* de destino.

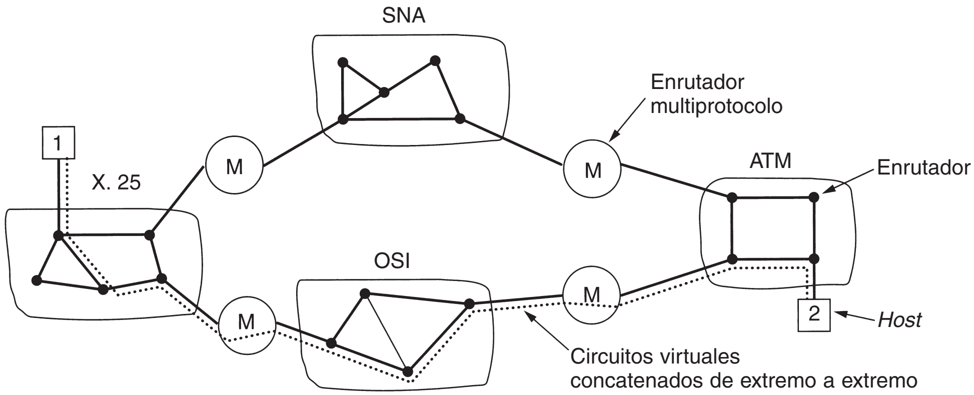


Figura 5-45. Interconectividad mediante circuitos virtuales concatenados.

Una vez que comienzan a fluir paquetes de datos por la ruta, cada puerta de enlace retransmite los paquetes de entrada y hace las conversiones entre los formatos de paquete y los números de circuito virtual, según sea necesario. Obviamente, todos los paquetes de datos deben atravesar la misma secuencia de puertas de enlace. En consecuencia, la red nunca reordena los paquetes de un flujo.

La característica esencial de este enfoque es que se establece una secuencia de circuitos virtuales desde el origen, a través de una o más puertas de enlace, hasta el destino. Cada puerta de enlace mantiene tablas que indican los circuitos virtuales que pasan a través suyo, a dónde se deben enrutar y el nuevo número de circuito virtual.

Este esquema funciona mejor cuando todas las redes tienen aproximadamente las mismas propiedades. Por ejemplo, si todas garantizan la entrega confiable de paquetes de capa de red, entonces, salvo una caída a lo largo de la ruta, el flujo del origen al destino también será confiable. De igual modo, si ninguna de ellas garantiza la entrega confiable, entonces la concatenación de los circuitos virtuales tampoco será confiable. Por otra parte, si la máquina de origen está en una red que garantiza la entrega confiable, pero una de las redes intermedias puede perder paquetes, la concatenación habrá cambiado fundamentalmente la naturaleza del servicio.

Los circuitos virtuales concatenados también son comunes en la capa de transporte. En particular, es posible construir un conducto de bits usando, digamos, SNA, que termine en una puerta de enlace, y luego tener una conexión TCP de esa puerta de enlace a la siguiente. De este modo, puede construirse un circuito virtual de extremo a extremo que abarque diferentes redes y protocolos.

5.5.4 Interconectividad no orientada a la conexión

El modelo alterno de interred es el modelo de datagramas, mostrado en la figura 5-46. En este modelo, el único servicio que ofrece la capa de red a la capa de transporte es la capacidad de inyectar datagramas en la subred y esperar que todo funcione bien. En la capa de red no hay noción en lo absoluto de un circuito virtual, y mucho menos de una concatenación de éstos. Este modelo no requiere que todos los paquetes que pertenecen a una conexión atraviesen la misma secuencia de puertas de enlace. En la figura 5-46 los datagramas del *host* 1 al *host* 2 toman diferentes rutas a través de la interred. Para cada paquete se toma una decisión de enrutamiento independiente, posiblemente dependiendo del tráfico en el momento del envío de dicho paquete. Esta estrategia puede utilizar múltiples rutas y lograr de esta manera un ancho de banda mayor que el modelo de circuitos virtuales concatenados. Por otra parte, no hay garantía de que los paquetes llegarán al destino en orden, suponiendo que lleguen.

El modelo de la figura 5-46 no es tan sencillo como parece. Por una parte, si cada red tiene su propio protocolo de capa de red, no es posible que un paquete de una red transite por otra. Podríamos imaginar a los enrutadores multiprotocolo tratando de traducir de un formato a otro, pero a menos que los dos formatos sean parientes cercanos con los mismos campos de información, tales conversiones siempre serán incompletas, y frecuentemente destinadas al fracaso. Por esta razón, pocas veces se intentan las conversiones.

Un segundo problema, más serio, es el direccionamiento. Imagine un caso sencillo: un *host* de Internet está tratando de enviar un paquete IP a un *host* en una red SNA adyacente. Se podría pensar en una conversión entre direcciones IP y SNA en ambas direcciones. Además, el concepto de lo que es direccionable es diferente. En IP, los *hosts* (en realidad las tarjetas de red) tienen direcciones. En SNA, entidades diferentes a los *hosts* (por ejemplo dispositivos de hardware) pueden también tener direcciones. En el mejor de los casos, alguien tendría que mantener una base de

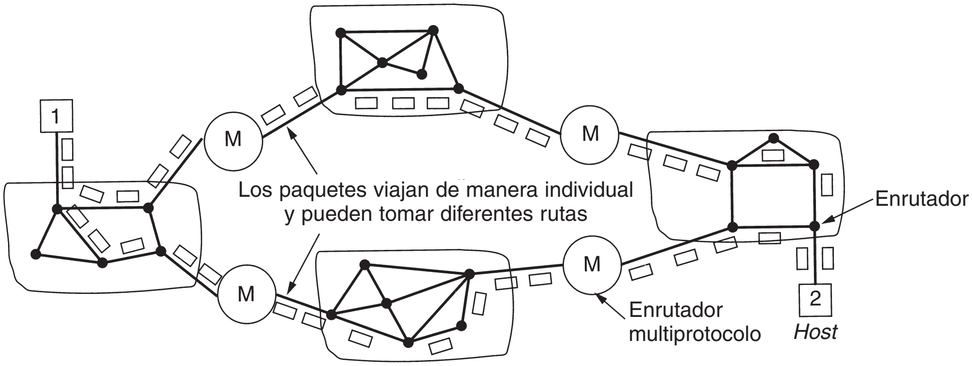


Figura 5-46. Una interred no orientada a la conexión.

datos de las conversiones de todo a todo en la medida de lo posible, pero esto sería constantemente una fuente de problemas.

Otra idea es diseñar un paquete universal de "interred" y hacer que todos los enrutadores lo reconozcan. Este enfoque es, precisamente, el que tiene IP: un paquete diseñado para llevarse por muchas redes. Por supuesto, podría suceder que IPv4 (el protocolo actual de Internet) expulse del mercado a todos los formatos, que Ipv6 (el protocolo futuro de Internet) no se vuelva popular y que no se invente nada más, pero la historia sugiere otra cosa. Hacer que todos se pongan de acuerdo en un solo formato es difícil, más aún cuando las empresas consideran que es una ventaja para ellas contar con un formato patentado bajo su control.

Recapitulemos ahora brevemente las dos maneras en que puede abordarse la interconectividad de redes. El modelo de circuitos virtuales concatenados tiene en esencia las mismas ventajas que el uso de circuitos virtuales en una sola subred: pueden reservarse búferes por adelantado, puede garantizarse la secuencia, pueden usarse encabezados cortos y pueden evitarse los problemas causados por paquetes duplicados retrasados.

El modelo también tiene las mismas desventajas: el espacio de tablas requerido en los enrutadores para cada conexión abierta, la falta de enrutamiento alterno para evitar áreas congestionadas y la vulnerabilidad a fallas de los enrutadores a lo largo de la ruta. También tiene la desventaja de que su implementación es difícil, si no imposible, si una de las redes que intervienen es una red no confiable de datagramas.

Las propiedades del enfoque por datagramas para la interconectividad son las mismas que las de las subredes de datagramas: un mayor potencial de congestión, pero también mayor potencial para adaptarse a él, la robustez ante fallas de los enrutadores y la necesidad de encabezados más grandes. En una interred son posibles varios algoritmos de enrutamiento adaptativo, igual que en una sola red de datagramas.

Una ventaja principal del enfoque por datagramas para la interconectividad es que puede usarse en subredes que no usan circuitos virtuales. Muchas LANs, redes móviles (por ejemplo, flotas

aéreas y navales) e incluso algunas WANs caen en esta categoría. Cuando una interred incluye una de éstas, surgen serios problemas si la estrategia de interredes se basa en circuitos virtuales.

5.5.5 Entunelamiento

El manejo del caso general de lograr la interacción de dos redes diferentes es en extremo difícil. Sin embargo, hay un caso especial común que puede manejarse. Este caso es cuando el *host* de origen y el de destino están en la misma clase de red, pero hay una red diferente en medio. Como ejemplo, piense en un banco internacional con una Ethernet basada en TCP/IP en París, una Ethernet basada en TCP/IP en Londres y una WAN no IP (por ejemplo, ATM) en medio, como se muestra en la figura 5-47.

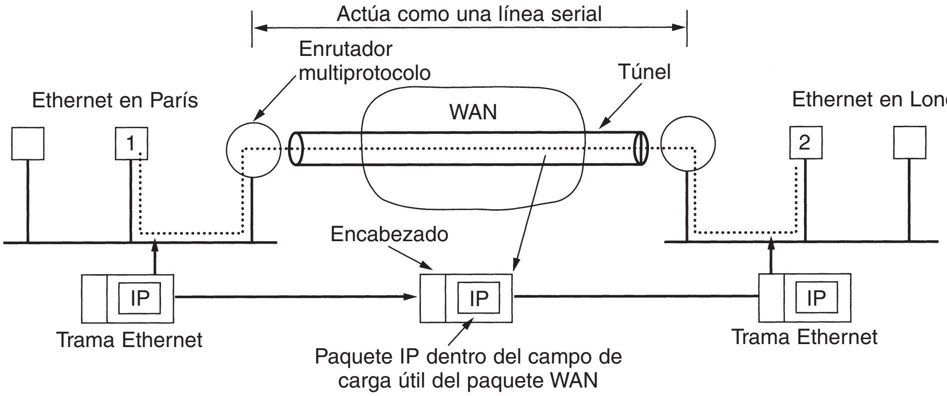


Figura 5-47. Entunelamiento de un paquete de París a Londres.

La solución a este problema es una técnica llamada **entunelamiento.** Para enviar un paquete IP al *host* 2, el *host* 1 construye el paquete que contiene la dirección IP del *host* 2, a continuación lo inserta en una trama Ethernet dirigida al enrutador multiprotocolo de París y, por ultimo, lo pone en la línea Ethernet. Cuando el enrutador multiprotocolo recibe la trama, retira el paquete IP, lo inserta en el campo de carga útil del paquete de capa de red de la WAN y dirige este último a la dirección de la WAN del enrutador multiprotocolo de Londres. Al llegar ahí, el enrutador de Londres retira el paquete IP y lo envía al *host* 2 en una trama Ethernet.

La WAN puede visualizarse como un gran túnel que se extiende de un enrutador multiprotocolo al otro. El paquete IP simplemente viaja de un extremo del túnel al otro, bien acomodado en una caja bonita. No tiene que preocuparse por lidiar con la WAN. Tampoco tienen que hacerlo los *hosts* de cualquiera de las Ethernets. Sólo el enrutador multiprotocolo tiene que entender los paquetes IP y WAN. De hecho, la distancia completa entre la mitad de un enrutador multiprotocolo y la mitad del otro actúa como una línea serial.

El entunelamiento puede aclararse mediante una analogía. Considere una persona que maneja su auto de París a Londres. En Francia, el auto se mueve con su propia energía, pero al llegar al Canal de la Mancha, se carga en un tren de alta velocidad y se transporta a Inglaterra a través del Chunnel (los autos no pueden conducirse a través del Chunnel). En efecto, el auto se transporta como carga, como se muestra en la figura 5-48. En el otro extremo, se libera el auto en las carreteras inglesas y nuevamente continúa moviéndose con sus propios medios. El entunelamiento de paquetes a través de una red foránea funciona de la misma manera.

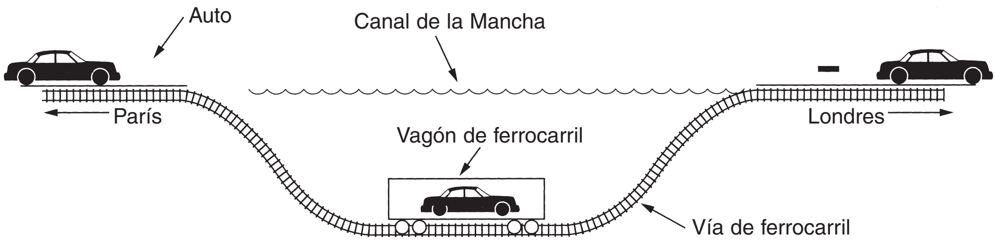


Figura 5-48. Paso de un auto de Francia a Inglaterra a través de un túnel.

5.5.6 Enrutamiento entre redes

El enrutamiento a través de una interred es parecido al enrutamiento en una sola subred, pero con algunas complicaciones adicionales. Por ejemplo, considere la interred de la figura 5-49(a) en la que cinco redes están conectadas mediante seis enrutadores (posiblemente multiprotocolo). Realizar un modelo de grafo de esta situación es complicado por el hecho de que cada enrutador multiprotocolo puede acceder (es decir, enviar paquetes) de manera directa a todos los demás enrutadores conectados a cualquier red a la que esté conectado. Por ejemplo, *B* en la figura 5-49(a) puede acceder directamente a *A* y a *C* a través de la red 2, y también a *D* a través de la red 3. Esto conduce al grafo de la figura 5-49(b).

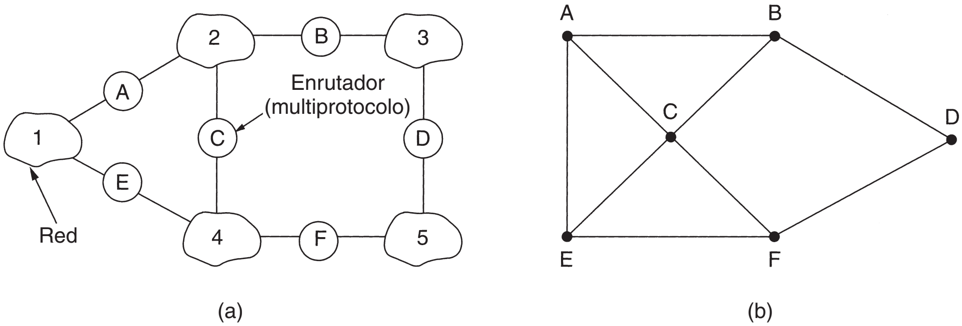


Figura 5-49. (a) Una interred. (b) Grafo de la interred.

Una vez construido el grafo, pueden aplicarse algoritmos de enrutamiento conocidos, como el algoritmo de vector de distancia y el de estado del enlace, al grupo de enrutadores multiprotocolo. Esto da un algoritmo de enrutamiento de dos niveles: en cada red se utiliza un **protocolo de puerta de enlace interior** (**IGP**), pero entre ellas se usa un **protocolo de puerta de enlace exterior** (**EGP**) ("puerta de enlace" es un término antiguo para "enrutador"). De hecho, debido a que estas redes son independientes, cada una puede utilizar un algoritmo diferente del de la otra. Puesto que cada red de una interred es independiente de las demás, con frecuencia se le llama **sistema autónomo** (**AS**).

Un paquete de interred típico parte de su LAN hacia el enrutador multiprotocolo local (en el encabezado de la capa de MAC). Al llegar ahí, el código de la capa de red decide por cuál enrutador multiprotocolo reenviará el paquete, usando sus propias tablas de enrutamiento. Si ese enrutador puede alcanzarse usando el protocolo de la red nativa del paquete, éste se reenvía directamente ahí. De otra manera, se envía por túnel, encapsulado en el protocolo requerido por la red que interviene. Este proceso se repite hasta que el paquete llega a la red de destino.

Una de las diferencias del enrutamiento entre las redes y el enrutamiento dentro de las redes es que el primero con frecuencia requiere el cruce de fronteras internacionales. De pronto, entran en escena varias leyes, como las estrictas leyes suecas de confidencialidad sobre la exportación de datos personales de ciudadanos suecos. Otro ejemplo es la ley canadiense que indica que el tráfico de datos que se origina en Canadá y llega a un destino en Canadá no puede dejar el país. Esto significa que el tráfico de Windsor, Ontario a Vancouver no puede enrutarse a través de Detroit, Estado Unidos, incluso si esta ruta es más rápida y barata.

Otra diferencia entre el enrutamiento interior y el exterior es el costo. Dentro de una sola red, normalmente se aplica un solo algoritmo de cargo. Sin embargo, redes diferentes pueden estar bajo administraciones diferentes, un una ruta puede ser menos cara que otra. Del mismo modo, la calidad de servicio ofrecida por diferentes redes puede ser distinta, y ésta puede ser una razón para escoger una ruta y no otra.

5.5.7 Fragmentación

Cada red impone un tamaño máximo a sus paquetes. Estos límites tienen varias razones, entre ellas:

1. El hardware (por ejemplo, el tamaño de una trama Ethernet).
2. El sistema operativo (por ejemplo, todos los búferes son de 512 bytes).
3. Los protocolos (por ejemplo, la cantidad de bits en el campo de longitud de paquete).
4. El cumplimiento de algún estándar (inter)nacional.
5. El deseo de reducir hasta cierto nivel las retransmisiones inducidas por errores.
6. El deseo de evitar que un paquete ocupe el canal demasiado tiempo.

El resultado de estos factores es que los diseñadores de redes no están en libertad de escoger cualquier tamaño máximo de paquetes que deseen. Las cargas útiles máximas van desde 48 bytes (celdas ATM) hasta 65,515 bytes (paquetes IP), aunque el tamaño de la carga útil en las capas superiores con frecuencia es más grande.

Surge un problema obvio cuando un paquete grande quiere viajar a través de una red cuyo tamaño máximo de paquete es demasiado pequeño. Una solución es asegurar que no ocurra el problema. En otras palabras, la interred debe usar un algoritmo de enrutamiento que evite el envío de paquetes a través de redes que no pueden manejarlos. Sin embargo, esta solución en realidad no es una solución. ¿Qué ocurre si el paquete original es demasiado grande para ser manejado por la red de destino? El algoritmo de enrutamiento no puede pasar por alto el destino.

Básicamente, la única solución al problema es permitir que las puertas de enlace dividan los paquetes en **fragmentos,** enviando cada paquete como paquete de interred individual. Sin embargo, como lo sabe cualquier padre de un niño pequeño, la conversión de un objeto grande en fragmentos pequeños es significativamente más fácil que el proceso inverso. (Los físicos incluso le han dado un nombre a este efecto: segunda ley de la termodinámica.) Las redes de conmutación de paquetes también tienen problemas al unir nuevamente los fragmentos.

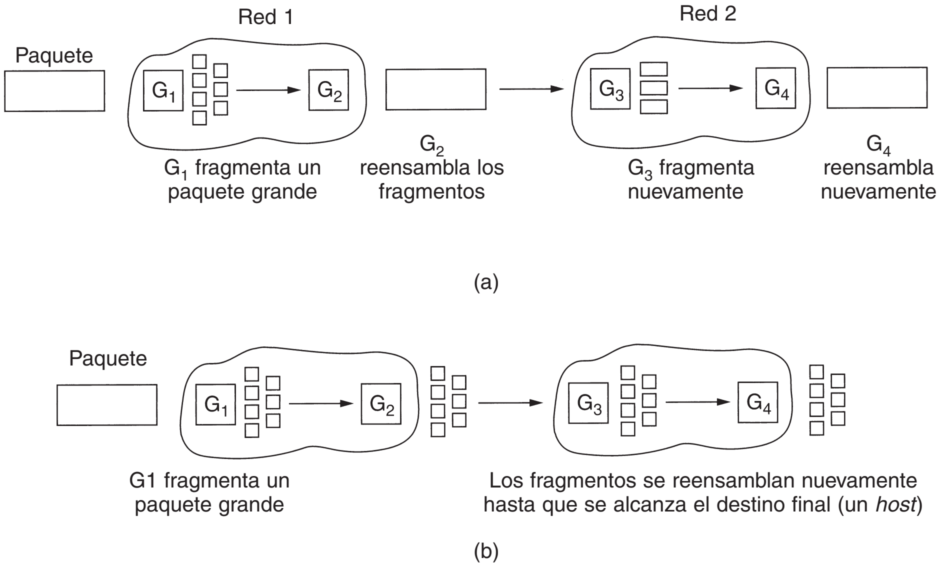


Figura 5-50. (a) Fragmentación transparente. (b) Fragmentación no transparente.

Existen dos estrategias opuestas para recombinar los fragmentos y recuperar el paquete original. La primera es hacer transparente la fragmentación causada por una red de "paquete pequeño" a las demás redes subsiguientes por las que debe pasar el paquete para llegar a su destino final. Esta opción se muestra en la figura 5-50(a). Con este método, la red de paquete pequeño tiene puertas de enlace (lo más probable es que sean enrutadores especializados) que interactúan con

otras redes. Cuando un paquete de tamaño excesivo llega a una puerta de enlace, ésta lo divide en fragmentos. Todos los fragmentos se dirigen a la misma puerta de enlace de salida, donde se recombinan las piezas. De esta manera se ha hecho transparente el paso a través de la red de paquete pequeño. Las redes subsiguientes ni siquiera se enteran de que ha ocurrido una fragmentación. Las redes ATM, por ejemplo, tienen *hardware* especial para proporcionar fragmentación transparente de paquetes en celdas y luego reensamblar las celdas en paquetes. En el mundo ATM, a la fragmentación se le llama segmentación; el concepto es el mismo, pero algunos de los detalles son diferentes.

La fragmentación transparente es sencilla, pero tiene algunos problemas. Por una parte, la puerta de enlace de salida debe saber cuándo ha recibido todas las piezas, por lo que debe incluirse un campo de conteo o un bit de "fin de paquete" en cada paquete. Por otra parte, todos los paquetes deben salir por la misma puerta de enlace. Al no permitir que algunos fragmentos sigan una ruta al destino final, y otros fragmentos una ruta distinta, puede bajar un poco el desempeño. Un último problema es la sobrecarga requerida para reensamblar y volver a fragmentar repetidamente un paquete grande que pasa a través de una serie de redes de paquete pequeño. ATM requiere fragmentación transparente.

La otra estrategia de fragmentación es abstenerse de recombinar los fragmentos en las puertas de enlace intermedias. Una vez que se ha fragmentado un paquete, cada fragmento se trata como si fuera un paquete original. Todos los fragmentos pasan a través de la puerta de enlace (o puertas de enlace) de salida, como se muestra en la figura 5-50(b). La recombinación ocurre sólo en el *host* de destino. IP funciona de esta manera.

La fragmentación no transparente también tiene algunos problemas. Por ejemplo, requiere que "*todos*" los *hosts* sean capaces de hacer el reensamble. Otro problema es que, al fragmentarse un paquete grande, aumenta la sobrecarga total, pues cada fragmento debe tener un encabezado. En tanto que en el primer método la sobrecarga desaparece en cuanto se sale de la red de paquete pequeño, en este método la sobrecarga permanece durante el resto de la travesía. Sin embargo, una ventaja de este método es que ahora pueden usarse varias puertas de enlace de salida, lográndose un mejor desempeño. Por supuesto, si se está usando el modelo de circuito virtual concatenado, esta ventaja no es de ninguna utilidad.

Cuando se divide un paquete, los fragmentos deben numerarse de tal manera que el flujo de datos original pueda reconstruirse. Una manera de numerar los fragmentos es usar un árbol. Si el paquete 0 debe dividirse, se llama a las partes 0.0, 0.1, 0.2, etcétera. Si estos mismos fragmentos deben fragmentarse después, las piezas se numeran como 0.0.0, 0.0.1, 0.0.2,...,0.1.0, 0.1.1, 0.1.2, etcétera. Si se reservan suficientes campos en el encabezado para el peor caso y no se generan duplicaciones en ningún lado, este esquema es suficiente para asegurar que todas las partes puedan reensamblarse correctamente en el destino, sin importar en qué orden lleguen.

Sin embargo, si cualquier red pierde o descarta paquetes, hay la necesidad de retransmisiones de extremo a extremo, con efectos poco afortunados para el sistema de numeración. Suponga que un paquete de 1024 bits se fragmenta inicialmente en cuatro fragmentos del mismo tamaño, 0.0, 0.1, 0.2 y 0.3. Se pierde el fragmento 0.1, pero las otras partes llegan al destino. En algún momento termina el temporizador del origen y se vuelve a transmitir el paquete original, pero esta vez la ruta tomada pasa a través de una red con un límite de 512 bits, por lo que se generan dos fragmentos.

Cuando el nuevo fragmento 0.1 llegue al destino, el receptor pensará que ya se han recibido las cuatro partes y reconstruirá incorrectamente el paquete.

Un sistema de numeración completamente diferente, y mejor, es que el protocolo de interred defina un tamaño de fragmento elemental lo bastante pequeño como para que el fragmento elemental pueda pasar a través de todas las redes. Al fragmentarse un paquete, todas las partes son iguales al tamaño de fragmento elemental, excepto la última, que puede ser más corta. Un paquete de interred puede contener varios fragmentos, por razones de eficiencia. El encabezado de interred debe proporcionar el número de paquete original y el número del (primer) fragmento elemental contenido en el paquete. Como siempre, también debe haber un bit que indique que el último fragmento elemental contenido en el paquete de interred es el último del paquete original.

Este método requiere dos campos de secuencia en el encabezado de interred: el número original de paquete y el número de fragmento. Evidentemente hay una concesión entre el tamaño del fragmento elemental y la cantidad de bits en el número de fragmento. Dado que se supone que el tamaño del fragmento elemental es aceptable para todas las redes, la fragmentación subsiguiente de un paquete de interred que contiene varios fragmentos no produce problemas. El límite último aquí es hacer que el fragmento elemental sea un solo bit o byte y, por lo tanto, el número de fragmento es el desplazamiento del bit o byte en el paquete original, como se muestra en la figura 5-51.

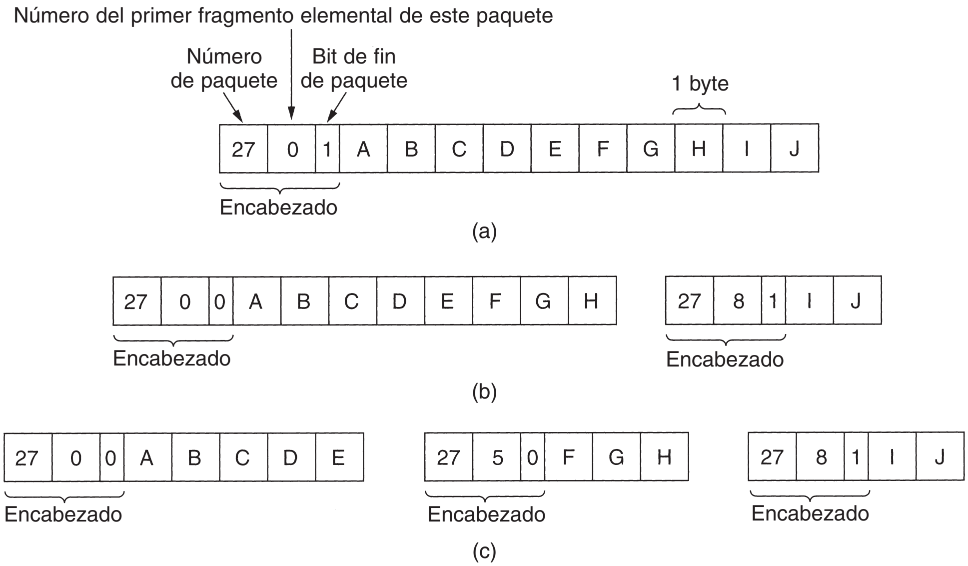


Figura 5-51. Fragmentación cuando el tamaño de datos elemental es de 1 byte. (a) Paquete original que contiene 10 bytes de datos. (b) Fragmentos tras pasar a través de una red con un tamaño máximo de paquete de 8 bytes de carga útil más encabezado. (c) Fragmentos tras pasar a través de una puerta de enlace de tamaño 5.

Algunos protocolos de interred llevan este método aún más lejos y consideran a toda la transmisión a través de un circuito virtual como un paquete gigantesco, por lo que cada fragmento contiene el número absoluto de byte del primer byte del fragmento.

5.6 LA CAPA DE RED DE INTERNET

Antes de entrar a los detalles específicos de la capa de red de Internet, vale la pena dar un vistazo a los principios que guiaron su diseño en el pasado y que hicieron posible el éxito que tiene hoy en día. En la actualidad, con frecuencia parece que la gente los ha olvidado. Estos principios se enumeran y analizan en el RFC 1958, el cual vale la pena leer (y debería ser obligatorio para todos los diseñadores de protocolos, así como un examen al final de la lectura). Este RFC utiliza mucho las ideas encontradas en (Clark, 1988, y Saltzer y cols., 1984). A continuación presentamos los que consideramos como los 10 mejores principios (del más al menos importante).

1. **Asegúrese de que funciona.** No termine el diseño o estándar hasta que múltiples prototipos se hayan comunicado entre sí de manera exitosa. Con demasiada frecuencia, los diseñadores primero escriben un estándar de 1000 páginas, lo aprueban y después descubren que tiene demasiadas fallas y no funciona. Después escriben la versión 1.1 de ese mismo estándar. Ésta no es la manera correcta de proceder.

2. **Mantenga la simplicidad.** Cuando tenga duda, utilice la solución más simple. William de Occam formuló este principio (la navaja de Occam) en el siglo XIV. Dicho en otras palabras: combata las características. Si una característica no es absolutamente esencial, descártela, especialmente si el mismo efecto se puede alcanzar mediante la combinación de otras características.

3. **Elija opciones claras.** Si hay varias maneras para realizar la misma tarea, elija sólo una. Tener dos o más formas de hacer lo mismo es buscarse problemas. Con frecuencia, los estándares tienen múltiples opciones o modos o parámetros debido a que personas poderosas insisten en que su método es el mejor. Los diseñadores deben resistir con fuerza esta tendencia. Simplemente diga no.

4. **Explote la modularidad.** Este principio lleva directamente a la idea de tener pilas de protocolos, cuyas capas son independientes entre sí. De esta forma, si las circunstancias requieren que un módulo o capa cambie, los otros no se verán afectados.

5. **Prevea la heterogeneidad.** En cualquier red grande habrán diferentes tipos de hardware, facilidades de transmisión y aplicaciones. Para manejarlos, el diseño de la red debe ser simple, general y flexible.

6. **Evite las opciones y parámetros estáticos.** Si los parámetros son inevitables (por ejemplo, el tamaño máximo del paquete), es mejor hacer que el emisor y el receptor negocien un valor que definir opciones fijas.

7. **Busque un buen diseño; no es necesario que sea perfecto.** Con frecuencia, los diseñadores tienen un buen diseño pero éste no puede manejar algún caso especial. En lugar de desbaratar el diseño, los diseñadores deberían dejar que esa parte la resuelvan las personas con el caso especial.
8. **Sea estricto cuando envíe y tolerante cuando reciba.** En otras palabras, sólo envíe paquetes que cumplan rigurosamente con los estándares, pero espere paquetes que tal vez no cumplan del todo y trate de lidiar con ellos.
9. **Piense en la capacidad de crecimiento.** Si el sistema va a manejar de manera efectiva millones de *hosts* y miles de millones de usuarios, las bases de datos no centralizadas de cualquier tipo son tolerables y la carga debe dispersarse lo más equitativamente posible en los recursos disponibles.
10. **Considere el desempeño y el costo.** Si una red tiene un desempeño pobre o un costo exagerado, nadie la utilizará.

Dejemos a un lado los principios generales y comencemos a ver los detalles de la capa de red de Internet. En la capa de red, la Internet puede verse como un conjunto de subredes, o **sistemas autónomos** interconectados. No hay una estructura real, pero existen varias redes dorsales principales. Éstas se construyen a partir de líneas de alto ancho de banda y enrutadores rápidos. Conectadas a las redes dorsales hay redes regionales (de nivel medio), y conectadas a estas redes regionales están las LANs de muchas universidades, compañías y proveedores de servicios de Internet. En la figura 5-52 se presenta un dibujo de esta organización cuasijerárquica.

El pegamento que mantiene unida a Internet es el protocolo de capa de red, **IP** (**Protocolo de Internet**)**.** A diferencia de la mayoría de los protocolos de capa de red anteriores, éste se diseñó desde el principio con la interconexión de redes en mente. Una buena manera de visualizar la capa de red es la siguiente. Su trabajo es proporcionar un medio de mejor esfuerzo (es decir, sin garantía) para el transporte de datagramas del origen al destino, sin importar si estas máquinas están en la misma red, o si hay otras redes entre ellas.

La comunicación en Internet funciona como sigue. La capa de transporte toma flujos de datos y los divide en datagramas. En teoría, los datagramas pueden ser de hasta 64 Kbytes cada uno, pero en la práctica por lo general son de unos 1500 bytes (por lo tanto, se ajustan en una trama de Ethernet). Cada datagrama se transmite a través de Internet, posiblemente fragmentándose en unidades más pequeñas en el camino. Cuando todas las piezas llegan finalmente a la máquina de destino, son reensambladas por la capa de red, dejando el datagrama original. A continuación este datagrama se entrega a la capa de transporte, que lo introduce en el flujo de entrada del proceso receptor. Como se muestra en la figura 5-52, un paquete que se origina en el *host* 1 tiene que atravesar seis redes para llegar al *host* 2. En la práctica, por lo general son más de seis.

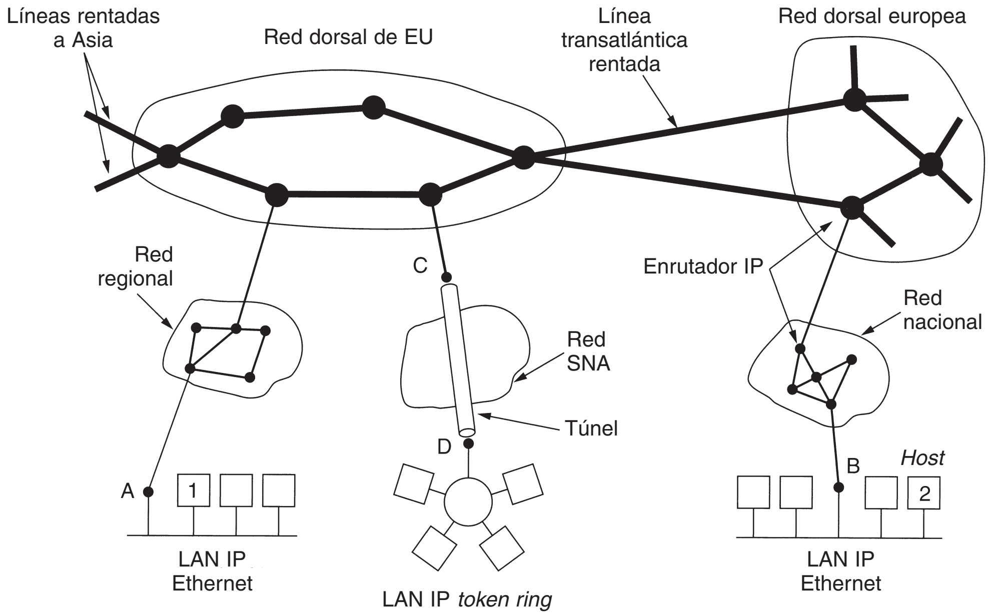


Figura 5-52. Internet es un conjunto interconectado de muchas redes.

5.6.1 El protocolo IP

Un lugar adecuado para comenzar nuestro estudio de la capa de red de Internet es el formato de los datagramas de IP mismos. Un datagrama IP consiste en una parte de encabezado y una parte de texto. El encabezado tiene una parte fija de 20 bytes y una parte opcional de longitud variable. El formato del encabezado se muestra en la figura 5-53. Se transmite en orden de *big endian*: de izquierda a derecha, comenzando por el bit de orden mayor del campo de *Versión*. (SPARC es *big endian*; Pentium es *little endian.*) En las máquinas *little endian*, se requiere conversión por software tanto para la transmisión como para la recepción.

El campo de *Versión* lleva el registro de la versión del protocolo al que pertenece el datagrama. Al incluir la versión en cada datagrama, es posible hacer que la transición entre versiones se lleve meses, o incluso años, ejecutando algunas máquinas la versión vieja y otras la versión nueva. En la actualidad, se está trabajando en una transición entre IPv4 e IPv6, la cual ha tomado años, y no está cerca de terminarse (Durand, 2001; Wiljakka, 2002, y Waddington y Chang, 2002). Algunas personas incluso piensan que nunca se terminará (Weiser, 2001). Como una adición a la numeración, IPv5 fue un protocolo de flujo experimental en tiempo real que no se utilizó ampliamente.

Dado que la longitud del encabezado no es constante, se incluye un campo en el encabezado, *IHL*, para indicar la longitud en palabras de 32 bits. El valor mínimo es de 5, cifra que se aplica cuando no hay opciones. El valor máximo de este campo de 4 bits es 15, lo que limita el encabezado

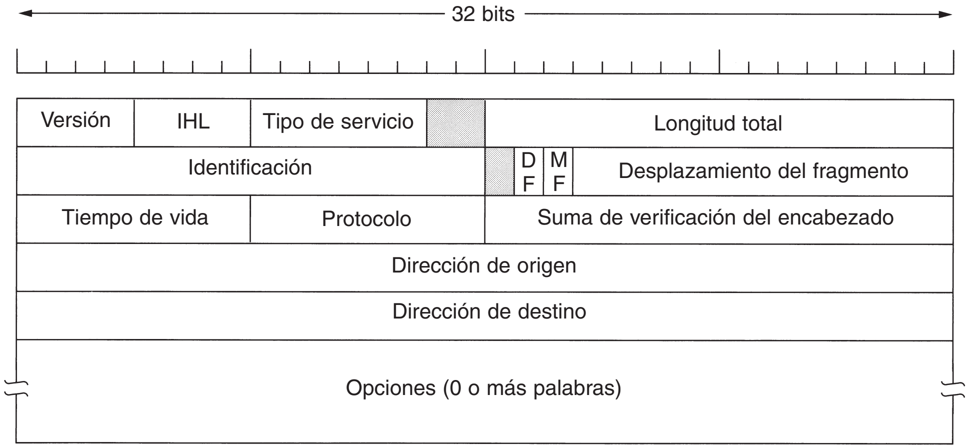


Figura 5-53. El encabezado de IPv4 (Protocolo Internet).

a 60 bytes y, por lo tanto, el campo de *Opciones* a 40 bytes. Para algunas opciones, por ejemplo para una que registre la ruta que ha seguido un paquete, 40 bytes es muy poco, lo que hace inútil esta opción.

El campo de *Tipo de servicio* es uno de los pocos campos que ha cambiado su significado (levemente) durante años. Su propósito aún es distinguir entre las diferentes clases de servicios. Son posibles varias combinaciones de confiabilidad y velocidad. Para voz digitalizada, la entrega rápida le gana a la entrega precisa. Para la transferencia de archivos, es más importante la transmisión libre de errores que la rápida.

Originalmente, el campo de 6 bits contenía (de izquierda a derecha) un campo de *Precedencia* de tres bits y tres banderas, *D*, *T* y *R*. El campo de *Precedencia* es una prioridad, de 0 (normal) a 7 (paquete de control de red). Los tres bits de bandera permiten al *host* especificar lo que le interesa más del grupo {retardo (*delay*), velocidad real de transporte (*throughput*), confiabilidad (*reliability*)}. En teoría, estos campos permiten a los enrutadores tomar decisiones entre, por ejemplo, un enlace satelital de alto rendimiento y alto retardo o una línea arrendada con bajo rendimiento y poco retardo. En la práctica, los enrutadores actuales ignoran por completo el campo de *Tipo de servicio,* a menos que se les indique lo contrario.

En algún momento, la IETF tiró la toalla y cambió el campo ligeramente para acomodar los servicios diferenciados. Seis de los bits se utilizan para indicar a cuáles de las clases de servicios analizadas anteriormente pertenece cada paquete. Estas clases incluyen las cuatro propiedades de encolamiento, tres posibilidades de eliminación y las clases históricas.

La *Longitud total* incluye todo el datagrama: tanto el encabezado como los datos. La longitud máxima es de 65,535 bytes. Actualmente este límite es tolerable, pero con las redes futuras de gigabits se requerirán datagramas más grandes.

El campo de *Identificación* es necesario para que el *host* de destino determine a qué datagrama pertenece un fragmento recién llegado. Todos los fragmentos de un datagrama contienen el mismo valor de *Identificación*.

A continuación viene un bit sin uso y luego dos campos de 1 bit. *DF* significa no fragmentar (*Don't Fragment*); es una orden para los enrutadores de que no fragmenten el datagrama, porque el destino es incapaz de juntar las piezas de nuevo. Por ejemplo, al arrancar una computadora, su ROM podría pedir el envío de una imagen de memoria a ella como un solo datagrama. Al marcar el datagrama con el bit *DF*, el transmisor sabe que llegará en una pieza, aún si significa que el datagrama debe evitar una red de paquete pequeño en la mejor ruta y tomar una ruta subóptima. Se requiere que todas las máquinas acepten fragmentos de 576 bytes o menos.

MF significa más fragmentos. Todos los fragmentos excepto el último tienen establecido este bit, que es necesario para saber cuándo han llegado todos los fragmentos de un datagrama.

El *Desplazamiento del fragmento* indica en qué parte del datagrama actual va este fragmento. Todos los fragmentos excepto el último del datagrama deben tener un múltiplo de 8 bytes, que es la unidad de fragmentos elemental. Dado que se proporcionan 13 bits, puede haber un máximo de 8192 fragmentos por datagrama, dando una longitud máxima de datagrama de 65,536 bytes, uno más que el campo de *Longitud total*.

El campo de *Tiempo de vida* es un contador que sirve para limitar la vida de un paquete. Se supone que este contador cuenta el tiempo en segundos, permitiendo una vida máxima de 255 seg; debe disminuirse en cada salto y se supone que disminuye muchas veces al encolarse durante un tiempo grande en un enrutador. En la práctica, simplemente cuenta los saltos. Cuando el contador llega a cero, el paquete se descarta y se envía de regreso un paquete de aviso al *host* de origen. Esta característica evita que los datagramas vaguen eternamente, algo que de otra manera podría ocurrir si se llegan a corromper las tablas de enrutamiento.

Una vez que la capa de red ha ensamblado un datagrama completo, necesita saber qué hacer con él. El campo de *Protocolo* indica el protocolo de las capas superiores al que debe entregarse el paquete. TCP es una posibilidad, pero también está UDP y algunos más. La numeración de los protocolos es global en toda Internet, y se define en el RFC 1700, pero en la actualidad dichos protocolos están contenidos en una base de datos en línea localizada en *www.iana.org*.

La *Suma de verificación del encabezado* verifica solamente el encabezado. Tal suma de verificación es útil para la detección de errores generados por palabras de memoria erróneas en un enrutador. El algoritmo es sumar todas las medias palabras de 16 bits a medida que llegan, usando aritmética de complemento a uno, y luego obtener el complemento a uno del resultado. Para los fines de este algoritmo, se supone que la *suma de verificación del encabezado* es cero cuando llega el paquete al destino. Este algoritmo es más robusto que una suma normal. Observe que la *suma de verificación del encabezado* debe recalcularse en cada salto, pues cuando menos uno de los campos siempre cambia (el campo de *Tiempo de vida*), pero pueden usarse trucos para acelerar el cálculo.

La *Dirección de origen* y la *Dirección de destino* indican el número de red y el número de *host*. Estudiaremos las direcciones de Internet en la siguiente sección. El campo de *Opciones* se diseñó para proporcionar un recurso que permitiera que las versiones subsiguientes del protocolo incluyeran información no presente en el diseño original, para permitir que los experimentadores prueben ideas nuevas y para evitar la asignación de bits de encabezado a información pocas veces necesaria. Las opciones son de longitud variable. Cada una empieza con un código de 1 byte que identifica la opción. Algunas opciones van seguidas de un campo de longitud de la opción de 1 byte, y luego de uno o más bytes de datos. El campo de *Opciones* se rellena para completar múltiplos de cuatro bytes. Originalmente se definieron cinco opciones, como se lista en la

figura 5-54, pero se han agregado otras más. La lista completa ahora se mantiene en línea en *www.iana.org/assignments/ip-parameters*

Opción	Descripción
Seguridad	Especifica qué tan secreto es el datagrama
Enrutamiento estricto desde el origen	Indica la ruta completa a seguir
Enrutamiento libre desde el origen	Da una lista de los enrutadores que no deben evitarse
Registrar ruta	Hace que cada enrutador agregue su dirección IP
Marca de tiempo	Hace que cada enrutador agregue su dirección y su marca de tiempo

Figura 5-54. Algunas de las opciones del IP.

La opción de *seguridad* indica qué tan secreta es la información. En teoría, un enrutador militar puede usar este campo para especificar que no se enrute a través de ciertos países que los militares consideren "malos". En la práctica, todos los enrutadores lo ignoran, por lo que su única función real es la de ayudar a los espías a encontrar la información importante con mayor facilidad.

La opción de *enrutamiento estricto desde el origen* da la ruta completa desde el origen hasta el destino como secuencia de direcciones IP. Se requiere que el datagrama siga esa ruta exacta. Esta opción se usa sobre todo cuando los administradores de sistemas envían paquetes de emergencia porque las tablas de enrutamiento se han dañado, o para hacer mediciones de tiempo.

La opción de *enrutamiento libre desde el origen* requiere que el paquete pase por los enrutadores indicados en la lista, y en el orden especificado, pero se le permite pasar a través de otros enrutadores en el camino. Normalmente, esta opción sólo indicará algunos enrutadores, para obligar a una ruta en particular. Por ejemplo, si se desea obligar a un paquete de Londres a Sydney a ir hacia el oeste en lugar de hacia el este, esta opción podría especificar enrutadores en Nueva York, Los Ángeles y Honolulú. Esta opción es de mucha utilidad cuando las consideraciones políticas o económicas dictan pasar a través de, o evitar, ciertos países.

La opción de *registrar ruta* indica a los enrutadores a lo largo de la ruta que agreguen su dirección IP al campo de opción. Esto permite a los administradores del sistema buscar fallas en los algoritmos de enrutamiento ("¿por qué todos los paquetes de Houston a Dallas pasan por Tokio primero?"). Al establecer inicialmente ARPANET, ningún paquete pasaba nunca por más de nueve enrutadores, por lo que 40 bytes de opciones eran más que suficientes. Como se mencionó antes, ahora esto es demasiado poco.

Por último, la opción de *marca de tiempo* es como la opción de *registrar ruta,* excepto que además de registrar su dirección IP de 32 bits, cada enrutador también registra una marca de tiempo de 32 bits. Esta opción también es principalmente para búsquedas de fallas en los algoritmos de enrutamiento.

5.6.2 Direcciones IP

Cada *host* y enrutador de Internet tiene una dirección IP, que codifica su número de red y su número de *host*. La combinación es única: no hay dos máquinas que tengan la misma dirección IP.

Todas las direcciones IP son de 32 bits de longitud y se usan en los campos de *Dirección de origen* y de *Dirección de destino* de los paquetes IP. En importante mencionar que una dirección IP realmente no se refiere a un *host*. En realidad se refiere a una interfaz de red, por lo que si un *host* está en dos redes, debe tener dos direcciones IP. Sin embargo, en la práctica, la mayoría de los *hosts* se encuentran en una red y, por lo tanto, tienen una dirección IP.

Por varias décadas, las direcciones IP se dividieron en cinco categorías, las cuales se listan en la figura 5-55. Esta asignación se ha llamado **direccionamiento con clase** (*classful addressing*). Ya no se utiliza, pero en la literatura aún es común encontrar referencias. Más adelante analizaremos el reemplazo del direccionamiento con clase.

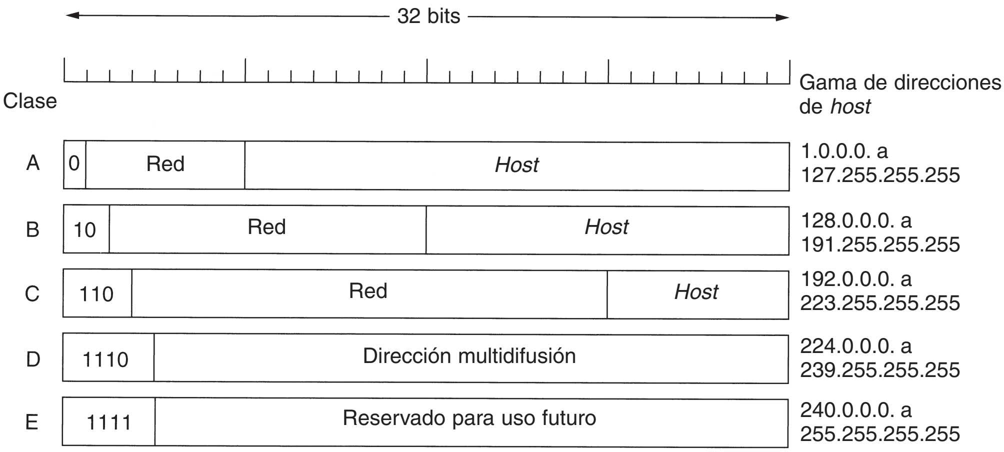


Figura 5-55. Formatos de dirección IP.

Los formatos de clase A, B, C y D permiten hasta 128 redes con 16 millones de *hosts* cada una, 16,382 redes de hasta 64K *hosts*, 2 millones de redes (por ejemplo, LANs) de hasta 256 *hosts* cada una (aunque algunas son especiales). También soportan la multidifusión, en la cual un datagrama es dirigido a múltiples *hosts*. Las direcciones que comienzan con 1111 se reservan para uso futuro. Hay cerca de 500,000 redes conectadas a Internet, y la cifra se duplica cada año. Los números de redes son manejados por una corporación no lucrativa llamada **ICANN** (**Corporación de Internet para la Asignación de Nombres y Números**) para evitar conflictos. A su vez, ICANN ha delegado partes del espacio de direcciones a varias autoridades regionales, las cuales han repartido direcciones IP a los ISPs y a otras compañías.

Las direcciones de red, que son números de 32 bits, generalmente se escriben en **notación decimal con puntos.** En este formato, cada uno de los 4 bytes se escribe en decimal, de 0 a 255. Por ejemplo, la dirección hexadecimal C0290614 se escribe como 192.41.6.20. La dirección IP menor es 0.0.0.0 y la mayor 255.255.255.255.

Los valores 0 y −1(todos 1s) tienen significado especial, como se muestra en la figura 5-56. El valor 0 significa esta red o este *host.* El valor −1 se usa como dirección de difusión para indicar todos los *hosts* de la red indicada.

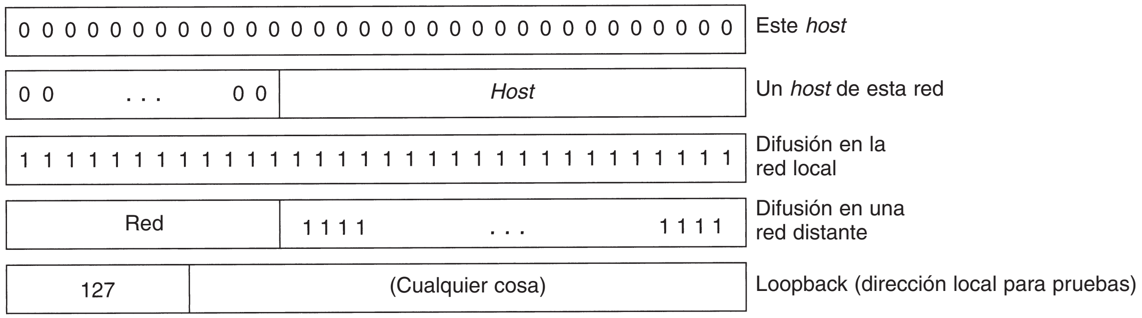


Figura 5-56. Direcciones IP especiales.

La dirección IP 0.0.0.0 es usada por los *hosts* cuando están siendo arrancados, pero no se usa después. Las direcciones IP con 0 como número de red se refieren a la red actual. Estas direcciones permiten que las máquinas se refieran a su propia red sin saber su número (pero tiene que saber su clase para saber cuántos 0s hay que incluir). La dirección que consiste solamente en 1s permite la difusión en la red local, por lo común una LAN. Las direcciones con un número de red propio y solamente unos en el campo de *host* permiten que las máquinas envíen paquetes de difusión a LANs distantes desde cualquier parte de Internet. Por último, todas las direcciones de la forma 127.*xx.yy.zz* se reservan para direcciones locales de prueba (*loopbacks*). Los paquetes enviados a esa dirección no se colocan en el cable; se procesan localmente y se tratan como paquetes de entrada. Esto permite que los paquetes se envíen a la red local sin que el transmisor conozca su número.

Subredes

Como hemos visto, todos los *hosts* de una red deben tener el mismo número de red. Esta propiedad del direccionamiento IP puede causar problemas a medida que crezcan las redes. Por ejemplo, considere una universidad que inició con una red de clase B utilizada por el Depto. de Ciencias de la Computación para las computadoras de su Ethernet. Un año más tarde, el Depto. de Ingeniería Eléctrica deseó conectarse a Internet, por lo que adquirió un repetidor para ampliar la Ethernet CS hasta su edificio. Conforme pasó el tiempo, muchos otros departamentos adquirieron computadoras y rápidamente se alcanzó el límite de cuatro repetidores por Ethernet. Se requirió una organización diferente.

Obtener una segunda dirección de red sería difícil debido a que las direcciones de red son escasas y la universidad ya tiene suficientes direcciones para aproximadamente 60,000 *hosts*. El problema es la regla de que una sola dirección de clase A, B o C haga referencia a una red, no a una colección de LANs. Conforme más y más organizaciones se encontraron en esta situación, se hizo un pequeño cambio al sistema de direccionamiento para manejar tal situación.

La solución a este problema es permitir la división de una red en varias partes para uso interno, pero aún actuar como una sola red ante el mundo exterior. En la actualidad, una red típica de un campus podría lucir como la que se muestra en la figura 5-57, con un enrutador principal conectado a un ISP o a una red regional, y numerosas Ethernets dispersas en diferentes departamentos

del campus. Cada una de las Ethernets tiene su propio enrutador conectado al enrutador principal (posiblemente mediante una LAN de red dorsal, pero la naturaleza de la conexión entre enrutadores no tiene relevancia aquí).

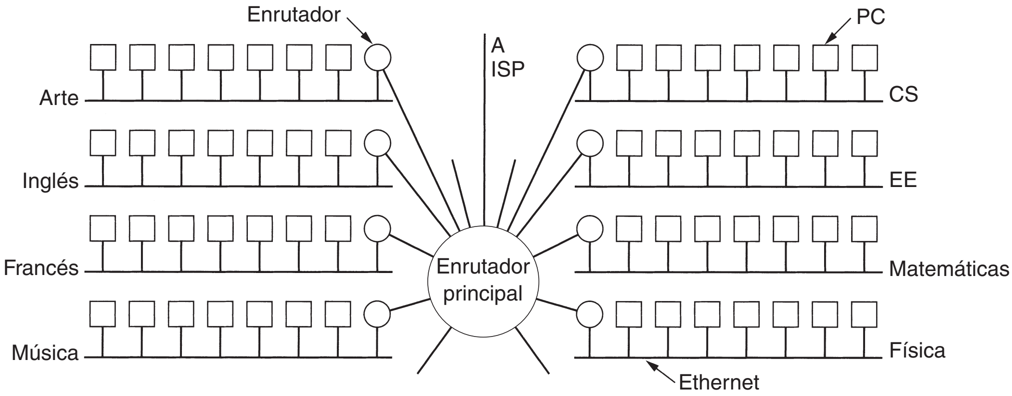


Figura 5-57. Una red de un campus que consiste de LANs para varios departamentos.

En la literatura sobre Internet, a estas partes de la red (en este caso Ethernets) se les llama **subredes.** Como mencionamos en el capítulo 1, este uso entra en conflicto con la "subred" cuyo significado es el grupo de todos los enrutadores y líneas de comunicación de una red. Esperamos que el contexto deje en claro el significado de que se trata. En esta y en la siguiente sección, la nueva definición será la que utilizaremos de manera exclusiva.

Cuando un paquete entra en el enrutador principal, ¿cómo sabe a cuál subred pasarlo (Ethernet)? Una forma sería tener una tabla con 65,536 entradas en el enrutador principal que indiquen cuál enrutador utilizar para cada *host* en el campus. Esta idea funcionaría, pero requeriría una tabla muy grande en el enrutador principal y mucho mantenimiento manual conforme se agregaran, movieran o eliminaran *hosts*.

En su lugar, se inventó un esquema diferente. Básicamente, en lugar de tener una sola dirección de clase B con 14 bits para el número de red y 16 bits para el número de *host*, algunos bits se eliminan del número de *host* para crear un número de subred. Por ejemplo, si la universidad tiene 35 departamentos, podría utilizar un número de subred de 6 bits y un número de *host* de 10 bits, lo que permitiría hasta 64 Ethernets, cada una con un máximo de 1022 *hosts* (0 y −1 no están disponibles, como se mencionó anteriormente). Esta división podría cambiarse posteriormente en caso de que no fuera correcta.

Para implementar subredes, el enrutador principal necesita una **máscara de subred** que indique la división entre el número de red + el número de subred y el *host*, como se muestra en la figura 5-58. Las máscaras de subred también se pueden escribir en notación decimal con puntos, o agregando a la dirección IP una diagonal seguida del número de bits usados para los números de red y subred. Para el ejemplo de la figura 5-58, la máscara de subred puede escribirse como 255.255.252.0. Una notación alternativa es /22 para indicar que la máscara de subred tiene una longitud de 22 bits.

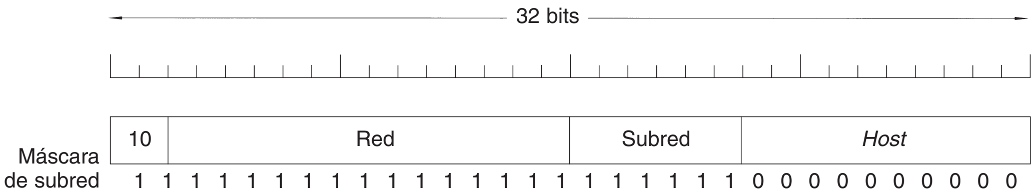


Figura 5-58. Una red de clase B dividida en 64 subredes.

Fuera de la red, la subred no es visible, por lo que la asignación de una subred nueva no requiere comunicación con el ICANN ni la modificación de bases de datos externas. En este ejemplo, la primera subred podría usar direcciones IP a partir de 130.50.4.1, la segunda podría empezar en 130.50.8.1, la tercera podría empezar en 130.50.12.1, etcétera. Para ver por qué las subredes se cuentan en grupos de cuatro, observe que las direcciones binarias correspondientes son como se muestra a continuación:

Subred 1: 10000010 00110010 000001|00 00000001
Subred 2: 10000010 00110010 000010|00 00000001
Subred 3: 10000010 00110010 000011|00 00000001

La barra vertical (|) muestra el límite entre el número de subred y el número de *host*. A su izquierda se encuentra el número de subred de 6 bits; a su derecha, el número de *host* de 10 bits.

Para ver el funcionamiento de las subredes, es necesario explicar la manera en que se procesan los paquetes IP en un enrutador. Cada enrutador tiene una tabla en la que se lista cierto número de direcciones IP (red, 0) y cierto número de direcciones IP (esta red, *host*). El primer tipo indica cómo llegar a redes distantes. El segundo tipo indica cómo llegar a redes locales. **La interfaz de red a utilizar para alcanzar el destino, así como otra información, está asociada a cada tabla.**

Cuando llega un paquete IP, se busca su dirección de destino en la tabla de enrutamiento. Si el paquete es para una red distante, se reenvía al siguiente enrutador de la interfaz dada en la tabla; si es para un *host* local (por ejemplo, en la LAN del enrutador), se envía directamente al destino. Si la red no está en la tabla, el paquete se reenvía a un enrutador predeterminado con tablas más extensas. Este algoritmo significa que cada enrutador sólo tiene que llevar el registro de otras redes y *hosts* locales (no de pares red-*host*), reduciendo en gran medida el tamaño de la tabla de enrutamiento.

Al introducirse subredes, se cambian las tablas de enrutamiento, agregando entradas con forma de (esta red, subred, 0) y (esta red, esta subred, *host*). Por lo tanto, un enrutador de la subred *k* sabe cómo llegar a todas las demás subredes y a todos los *hosts* de la subred *k*; no tiene que saber los detalles sobre los *hosts* de otras subredes. De hecho, todo lo que se necesita es hacer que cada enrutador haga un AND booleano con la máscara de subred de la red para deshacerse del número de *host* y buscar la dirección resultante en sus tablas (tras determinar de qué clase de red se trata).

Por ejemplo, a un paquete dirigido a 130.50.15.6 que llega a un enrutador de la subred 5 se le aplica un AND con la máscara de subred 255.255.252.0/22 para dar la dirección 130.50.12.0. Esta dirección se busca en las tablas de enrutamiento para averiguar la manera de llegar a los *hosts* de la subred 3. Por lo tanto, la división de redes reduce espacio en la tabla de enrutamiento creando una jerarquía de tres niveles, que consiste en red, subred y *host*.

CIDR—Enrutamiento interdominios sin clases

El IP ya ha estado en uso intensivo por más de una década; ha funcionado extremadamente bien, como lo demuestra el crecimiento exponencial de la Internet. Desgraciadamente, el IP se está convirtiendo con rapidez en víctima de su propia popularidad: se le están acabando las direcciones. Este desastre inminente ha propiciado una gran cantidad de controversias y debates en la comunidad de Internet sobre lo que debe hacerse al respecto. En esta sección describiremos tanto el problema como varias soluciones propuestas.

En 1987 unos pocos visionarios predijeron que algún día Internet podría crecer hasta 100,000 redes. La mayoría de los expertos minimizaron el problema aduciendo que ocurriría en muchas décadas, si es que llegaba a ocurrir. En 1996 se conectó la red 100,000. El problema, en pocas palabras, es que Internet se está quedando rápidamente sin direcciones de IP. En teoría, existen cerca de dos mil millones de direcciones, pero la práctica de organizar el espacio de direcciones por clases (véase la figura 5-55) desperdicia millones de ellas. En particular, el verdadero villano es la red clase B. Para la mayoría de las organizaciones, una red clase A, con 16 millones de direcciones, es demasiado grande, y una red clase C, de 256 direcciones, es demasiado pequeña. Una red clase B, con 65,536, es la adecuada. En el folclor de Internet, esta situación se conoce como el **problema de los tres osos** (del cuento *Ricitos de Oro y los tres osos*).

En realidad, una dirección clase B es demasiado grande para la mayoría de las organizaciones. Hay estudios que demuestran que más de la mitad de todas las redes clase B tienen menos de 50 *hosts*. Una red clase C habría bastado, pero sin duda todas las organizaciones que solicitaron una dirección clase B pensaron que un día el campo de *hosts* de 8 bits les quedaría pequeño. En retrospectiva, podría haber sido mejor que las redes clase C usaran 10 bits en lugar de 8 para el número de *host,* permitiendo 1022 *hosts* por red. De haber sido éste el caso, la mayoría de las organizaciones probablemente se habría conformado con una red clase C, y habría habido medio millón de ellas (en vez de sólo 16,384 redes clase B).

Es duro culpar a los diseñadores de Internet por no haber proporcionado más (y más pequeñas) direcciones de clase B. En el momento en que se tomó la decisión de crear las tres clases, Internet era una red de investigación que conectaba las principales universidades de investigación en Estados Unidos (más un número muy pequeño de compañías y sitios del ejército que hacían investigación de redes). En esa época nadie percibía que Internet llegaría a ser un sistema de comunicación de mercado masivo que rivalizaría con la red telefónica. También en esa época alguien dijo sin dudar: “Estados Unidos tiene alrededor de 2000 universidades. Aun cuando todas se

conectaran a Internet y muchas universidades en otros países también, nunca llegaremos a 16,000 puesto que no hay tantas universidades en todo el mundo. Además, como el número de *host* es un número integral de bytes acelera el procesamiento del paquete".

Sin embargo, si la división hubiera asignado 20 bits al número de red de clase B, habría surgido más rápidamente otro problema: la explosión de tablas de enrutamiento. Desde el punto de vista de los enrutadores, el espacio de direcciones IP es una jerarquía de dos niveles, con números de red y números de *host*. Los enrutadores no tienen que saber de todos los *hosts*, pero sí tienen que saber de todas las redes. Si se usaran medio millón de redes de clase C, todos los enrutadores de Internet necesitarían una tabla con medio millón de entradas, una por red, indicando la línea a usar para llegar a esa red, así como otra información.

Actualmente, el almacenamiento de medio millón de entradas tal vez es posible, aunque caro en los enrutadores críticos que guardan las tablas en la RAM estática de las tarjetas de E/S. Un problema más serio es que la complejidad de diferentes algoritmos relacionados con el mantenimiento de tablas crece con una tasa mayor que la lineal. Pero aún, mucho del *software* y *firmware* existente de los enrutadores se diseñó en un momento en que Internet tenía 1000 redes conectadas, y tener 10,000 redes parecía estar a décadas de distancia. Las decisiones de diseño tomadas entonces ya no tienen nada de óptimas.

Además, diversos algoritmos de enrutamiento requieren que cada enrutador transmita sus tablas periódicamente (por ejemplo, protocolos de vector de distancia). Cuanto más grandes sean las tablas, más probabilidad habrá de que algunas partes se pierdan en el camino, dando pie a datos incompletos en el otro lado y posiblemente a inestabilidades de enrutamiento.

El problema de las tablas de enrutamiento podría haberse resuelto usando una jerarquía más. Por ejemplo, podría haber servido hacer que cada dirección IP tuviera un campo de país, estado, ciudad, red y *host*. Entonces cada enrutador sólo necesitaría saber cómo llegar a los países, a los estados o provincias de su propio país, a las ciudades de su estado o provincia y a las redes de su ciudad. Por desgracia, esta solución requeriría bastante más de 32 bits para las direcciones de IP y usaría ineficientemente las direcciones (Liechtenstein tendría tantos bits como Estados Unidos).

En resumen, la mayoría de las soluciones resuelven un problema pero crean uno nuevo. La solución que se implementó y que dio a Internet un respiro es el **CIDR** (**Enrutamiento Interdominios sin Clases**). El concepto básico del CIDR, que se describe en el RFC 1519, es asignar las direcciones IP restantes en bloques de tamaño variable, independientemente de las clases. Si un sitio necesita, digamos, 2000 direcciones, se le da un bloque de 2048 direcciones con un límite de 2048 bytes.

Eliminar las clases hace más complicado el reenvío. En el sistema antiguo con clases el reenvío se hacía de la siguiente manera: cuando un paquete llegaba a un enrutador, una copia de la dirección IP se desplazaba 28 bits a la derecha para obtener un número de clase de 4 bits. Entonces una rama de 16 vías ordenaba los paquetes en A, B, C y D (si lo soportaba), con ocho de los casos para la clase A, cuatro de los casos para la clase B, dos de los casos para la clase C, uno para D y otro para E. A continuación, el código para cada clase enmascaraba los números de red de 8, 16 o 24 bits y los alineaba a la derecha en una palabra de 32 bits. Entonces se buscaba el número de la red en la tabla A, B o C, por lo común mediante indexación en las redes A y B y apli-

cando *hash* en las redes C. Una vez que se encontrara la entrada, se podría buscar la línea de salida y remitir el paquete.

Con CIDR, este algoritmo sencillo ya no funciona. En cambio, cada entrada de tabla de enrutamiento se extiende para darle una máscara de 32 bits. De esta manera, ahora hay una sola tabla de enrutamiento para todas las redes que consten de un arreglo de tres variables (dirección IP, máscara de subred, línea saliente). Cuando llega un paquete, primero se extrae su dirección de destino IP. Luego (conceptualmente) se analiza la tabla de enrutamiento entrada por entrada, enmascarando la dirección de destino y comparándola con la entrada de la tabla buscando una correspondencia. Es posible que coincidan entradas múltiples (con diferentes longitudes de máscara de subred), en cuyo caso se usa la máscara más larga. De esta manera, si hay una coincidencia para una máscara /20 y una máscara /24, se usa la entrada /24.

Se han ideado algoritmos complejos para acelerar el proceso de coincidencia de dirección (Ruiz-Sánchez y cols., 2001). Los enrutadores comerciales usan chips VLSI programados con estos algoritmos.

Para hacer más claro este proceso de comparación, consideremos un ejemplo en el cual se dispone de millones de direcciones, empezando en 194.24.0.0. Suponga que la Universidad de Cambridge necesita 2048 direcciones y se le asignan las direcciones 194.24.0.0 a 194.24.7.255, junto con la máscara 255.255.248.0. Enseguida, la Universidad de Oxford solicita 4096 direcciones. Puesto que un bloque de 4096 direcciones debe caer en un límite de 4096 bytes, no pueden asignarse las direcciones que comienzan en 194.24.8.0. En cambio, Oxford recibe 194.24.16.0 a 194.24.31.255, junto con la máscara de subred 255.255.240.0. Ahora la Universidad de Edimburgo solicita 1024 direcciones, y se le asignan las direcciones 194.24.8.0 a 194.24.11.255 y la máscara 255.255.252.0. Estas asignaciones se resumen en la figura 5.59.

Universidad	Primera dirección	Segunda dirección	Cantidad	Escrito como
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edimburgo	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Disponible)	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

Figura 5-59. Un conjunto de asignaciones de direcciones IP.

Las tablas de enrutamiento de todo el mundo se actualizan con tres entradas, que contienen una dirección base y una máscara de subred. Estas entradas (en binario) son:

```
   Dirección                                Máscara
C: 11000010 00011000 00000000 00000000     11111111 11111111 11111000 00000000
E: 11000010 00011000 00001000 00000000     11111111 11111111 11111100 00000000
O: 11000010 00011000 00010000 00000000     11111111 11111111 11110000 00000000
```

Ahora considere lo que ocurre cuando llega un paquete dirigido a 194.24.17.4, que en binario se representa con la siguiente cadena de 32 bits:

11000010 00011000 00010001 00000100

Primero se le hace un AND booleano con la máscara de Cambridge para obtener

11000010 00011000 00010000 00000000

Este valor no es igual a la dirección base de Cambridge, por lo que vuelve a hacerse un AND con la máscara de Edimburgo para obtener

11000010 00011000 00010000 00000000

Este valor no coincide con la dirección base de Edimburgo, por lo que se intenta con Oxford, obteniendo

11000010 00011000 00010000 00000000

Este valor coincide con la base de Oxford. Si no se encuentran más correspondencias recorriendo la tabla, se usa la entrada Oxford y se envía el paquete junto con la línea denominada en él.

Ahora veamos estas tres universidades desde el punto de vista de un enrutador en Omaha, Nebraska que tiene sólo cuatro líneas de salida: Minneapolis, Nueva York, Dallas y Denver. Cuando el software del enrutador tiene las tres nuevas entradas ahí, observa que puede combinar las tres entradas en una sola **entrada agregada** 194.24.0.0/19 con una dirección binaria y una submáscara como sigue:

11000010 0000000 00000000 00000000   11111111 11111111 11100000 00000000

Esta entrada envía todos los paquetes destinados para cualquiera de las tres universidades de Nueva York. Agregando las tres entradas, el enrutador de Omaha ha reducido en dos entradas el tamaño de su tabla.

Si Nueva York tiene una sola línea a Londres para todo el tráfico del Reino Unido, también puede usar una entrada agregada. Sin embargo, si tiene líneas separadas para Londres y Edimburgo, entonces debe tener tres entradas separadas. La agregación se usa en gran medida en Internet para reducir el tamaño de las tablas de enrutador.

Como una nota final a este ejemplo, la entrada de ruta agregada en Omaha envía también a Nueva York paquetes para las direcciones no asignadas. Si en verdad las direcciones no están asignadas esto no afecta, porque no se supone que ocurran. Sin embargo, si después se asignan a una compañía en California, se necesitará una entrada adicional, 194.24.12.0/22, para estas direcciones.

NAT—Traducción de Dirección de Red

Las direcciones IP son escasas. Un ISP podría tener una dirección de /16 (anteriormente de clase B), dándole 65,534 números de *host*. Si tiene más clientes que esos, tiene un problema. Para

clientes propios con las conexiones de línea conmutada, una manera de resolver el problema es asignar dinámicamente una dirección IP a una computadora cuando ésta llama e inicia la sesión y tomar de vuelta la dirección IP cuando se termina la sesión. De esta manera, una sola dirección de /16 puede manejar hasta 65,534 usuarios activos lo que es probablemente bastante bueno para un ISP con varios cientos de miles de clientes. Cuando termina la sesión, la dirección IP se reasigna a otra visita. Mientras esta estrategia trabaja bien para un ISP con un número moderado de usuarios propios, falla para ISPs que sirven sobre todo a clientes comerciales.

El problema es que los clientes comerciales esperan estar en línea continuamente durante las horas hábiles. Tanto los negocios pequeños —por ejemplo, agencias de viaje de tres personas— como las corporaciones grandes tienen varias computadoras conectadas por una LAN. Algunas computadoras son PCs de empleados; otras pueden ser servidores Web. Generalmente hay un enrutador en la LAN que se conecta al ISP por una línea rentada para proporcionar conectividad continua. Este arreglo significa que cada computadora debe tener todo el día mucho tiempo su propia dirección IP. De hecho, el número total de computadoras poseído por todos sus clientes comerciales combinados no puede exceder el número de direcciones IP que el ISP tiene. Para una dirección de /16, esto limita el número total de computadoras a 65,534. Para un ISP con decenas de miles de clientes comerciales, este límite se excederá rápidamente.

Para empeorar las cosas, más y más usuarios caseros se suscriben a ADSL o Internet por cable. Dos de los rasgos de estos servicios son: (1) el usuario consigue una dirección IP permanente y (2) no hay ningún cargo por la conexión (sólo un pago fijo mensual), por lo que los usuarios de ADSL y de cable se quedan registrados de manera permanente. Este desarrollo se agrega a la escasez de direcciones IP. Asignar direcciones IP conforme se solicitan (dinámicamente) como se hace con los usuarios de conexión por línea conmutada es inútil porque en cualquier momento el número de direcciones IP en uso puede ser muchas veces el número que el ISP posee.

Y sólo para hacerlo un poco más complicado, muchos ADSL y usuarios de cable tienen dos o más computadoras en casa, a menudo una para cada integrante de la familia y todos quieren estar en línea todo el tiempo usando la única dirección IP que su ISP les ha dado. La solución aquí es conectar todas las PCs a través de una LAN y poner un enrutador. Desde el punto de vista del ISP, ahora la familia se parece a un negocio comercial pequeño con un puñado de computadoras. Bienvenido a Pérez, Inc.

El problema de quedarse sin direcciones IP no es un problema teórico que podría ocurrir en algún punto en el futuro distante. Está sucediendo aquí y ahora mismo. La solución a largo plazo es que todo Internet emigre a IPv6, que tiene direcciones de 128 bits. Esta transición se está dando despacio, pero todo el proceso estará completo en cuestión de años. Como consecuencia, algunas personas sentían que se necesitaba un arreglo rápido a corto plazo. Este arreglo surgió en la forma de la **Traducción de Dirección de Red** (**NAT**) que se describe en el RFC 3022 y que resumiremos más adelante. Para información adicional, vea (Dutcher, 2001).

La idea básica de NAT es asignar una sola dirección IP a cada compañía (o a lo sumo, un número pequeño) para el tráfico de Internet. *Dentro* de la compañía, cada computadora tiene una dirección IP única que se usa para enrutar el tráfico interno. Sin embargo, cuando un paquete sale de la compañía y va al ISP, se presenta una traducción de dirección. Para hacer posible este esquema

los tres rangos de direcciones IP se han declarado como privados. Las compañías pueden usarlos internamente cuando lo deseen. La única regla es que ningún paquete que contiene estas direcciones puede aparecer en la propia Internet. Los tres rangos reservados son:

10.0.0.0	– 10.255.255.255/8	(16,777,216 *hosts*)
172.16.0.0	– 172.31.255.255/12	(1,048,576 *hosts*)
192.168.0.0	– 192.168.255.255/16	(65,536 *hosts*)

El primer rango proporciona 16,777,216 direcciones (excepto 0 y −1, como de costumbre) y es la opción usual de la mayoría de las compañías, incluso si no necesitan tantas direcciones.

El funcionamiento de NAT se muestra en la figura 5-60. Dentro de las instalaciones de la compañía, cada máquina tiene una dirección única de la forma 10.*x.y.z.* Sin embargo, en este ejemplo cuando un paquete sale de las instalaciones de la compañía, pasa a través de una **caja NAT** que convierte la dirección interna de origen de IP, 10.0.0.1 en la figura, a la verdadera dirección IP de la compañía, 198.60.42.12. A menudo, la caja NAT se combina en un solo dispositivo con un *firewall* (servidor de seguridad) que proporciona seguridad controlando cuidadosamente lo que entra y sale de la compañía. En el capítulo 8 estudiaremos los servidores de seguridad. También es posible integrar la caja NAT en el enrutador de la compañía.

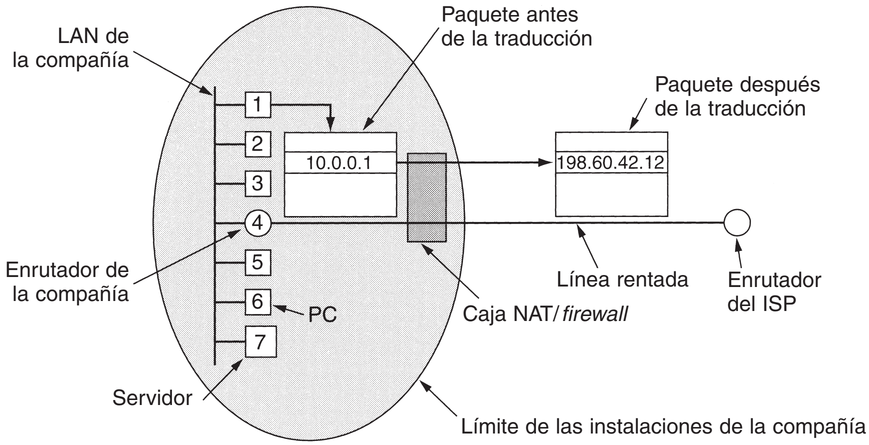


Figura 5-60. Colocación y funcionamiento de una caja NAT.

Hasta ahora hemos ignorado un pequeño detalle: cuando la respuesta vuelve (por ejemplo, un servidor Web), se dirige naturalmente a 198.60.42.12, por lo que, ¿cómo sabe ahora la caja NAT con qué dirección se reemplaza? Aquí está el problema con NAT. Si hubiera un campo de repuesto en el encabezado IP, ese campo podría usarse para guardar el registro del emisor real, pero sólo queda 1 bit sin usar. En principio, podría crearse una nueva opción para mantener la verdadera dirección del origen, pero haciéndolo así se requeriría cambiar el código IP en todas las máquinas de todo Internet para manejar la nueva opción. Ésta no es una alternativa prometedora para un arreglo rápido.

Lo que realmente pasó es lo siguiente. Los diseñadores de NAT observaron que la mayoría de paquetes IP lleva cargas útiles de TCP o UDP. Cuando estudiemos TCP y UDP en el capítulo 6, veremos que los dos tienen encabezados que contienen un puerto de origen y un puerto de destino. Más adelante explicaremos los puertos TCP, pero esa misma explicación es válida para los puertos UDP. Los puertos son enteros de 16 bits que indican dónde empieza y dónde acaba la conexión TCP. Estos puertos proporcionan el campo requerido para hacer que NAT funcione.

Cuando un proceso quiere establecer una conexión TCP con un proceso remoto, se conecta a un puerto TCP sin usar en su propia máquina. Éste se conoce como **puerto de origen** y le indica al código TCP codifican dónde enviar paquetes entrantes que pertenecen a esta conexión. El proceso también proporciona un **puerto de destino** para decir a quién dar los paquetes en el lado remoto. Los puertos 0-1023 se reservan para servicios bien conocidos. Por ejemplo, 80 es el puerto usado por los servidores Web, para que los clientes remotos puedan localizarlos. Cada mensaje TCP saliente contiene un puerto de origen y un puerto de destino. Juntos, estos puertos sirven para identificar los procesos que usan la conexión en ambos extremos.

Una analogía aclara el uso de los puertos. Imagine una compañía con un solo número de teléfono principal. Cuando las personas llaman al número principal, se encuentran con un operador que pregunta qué extensión quieren y entonces los ponen en esa extensión. El número principal es análogo a la dirección IP de la compañía y las extensiones en ambos extremos son análogas a los puertos. Los puertos son de 16 bits extra de dirección que identifican qué proceso obtiene cuál paquete entrante.

Usando el campo *Puerto de origen*, podemos resolver nuestro problema de conversión. Siempre que un paquete saliente entra en la caja NAT, la dirección de origen 10.*x*.*y*.*z* se reemplaza por la verdadera dirección IP de la compañía. Además, el campo *Puerto de origen* TCP se reemplaza por un índice en la tabla de traducción de la entrada 65,536 de la caja NAT. Esta entrada de la tabla contiene el puerto de origen y la dirección IP originales. Finalmente, las sumas de verificación de los encabezados IP y TCP se recalculan e insertan en el paquete. Es necesario reemplazar el *Puerto de origen* porque podría ocurrir que ambas conexiones de las máquinas 10.0.0.1 y 10.0.0.2 usaran el puerto 5000, por ejemplo, así que el *Puerto de origen* no basta para identificar el proceso de envío.

Cuando un paquete llega a la caja NAT desde el ISP, el *Puerto de origen* en el encabezado TCP se extrae y utiliza como un índice en la tabla de traducción de la caja NAT. Desde la entrada localizada, la dirección IP interna y el *Puerto de origen* TCP se extraen e insertan en el paquete. Luego, las sumas de verificación de IP y TCP se recalculan e insertan en el paquete. Entonces el paquete se pasa al enrutador de la compañía para su entrega normal utilizando la dirección 10.*x*.*y*.*z*.

También puede usarse NAT para aliviar la escasez de IP para usuarios de cable y ADSL. Cuando el ISP le asigna una dirección a cada usuario, usa direcciones 10.*x*.*y*.*z*. Cuando los paquetes de máquinas de usuario salen del ISP y entran en la Internet principal, atraviesan una caja NAT que los traduce a la verdadera dirección de Internet del ISP. En el camino de regreso, los paquetes sufren la conversión inversa. En este caso, para el resto de Internet, el ISP y sus usuarios caseros de cable y ADSL son como una compañía grande.

Aunque esta clase de esquema resuelve el problema, muchas personas en la comunidad de IP lo consideran a primera vista como una abominación. Brevemente resumidas, aquí están algunas de las objeciones. Primero, NAT viola el modelo arquitectónico de IP que establece que cada dirección IP identifica una sola máquina globalmente. Toda la estructura del software de Internet se basa en este hecho. Con NAT, las miles de máquinas pueden usar (y lo hacen) la dirección 10.0.0.1.

Segundo, NAT cambia a Internet de una red sin conexión a un tipo de red orientada a la conexión. El problema es que la caja NAT debe mantener la información (la conversión) para cada conexión que la atraviesa. Mantener el estado de la conexión es una propiedad de las redes orientadas a la conexión, no de las no orientadas. Si la caja NAT se cae y se pierde su tabla de traducción, todas sus conexiones TCP se destruyen. En ausencia de NAT, la destrucción de los enrutadores no afecta al TCP. El proceso de envío apenas queda fuera unos segundos y retransmite todos los paquetes cuya recepción no se haya confirmado. Con NAT, Internet es tan vulnerable como una red de circuitos conmutados.

Tercero, NAT viola la regla más fundamental de los protocolos de capas: la capa *k* no puede hacer ninguna suposición de en qué capa k + 1 ha puesto el campo de carga útil. Este principio básico está ahí para mantener independientes las capas. Si TCP se actualiza después a TCP-2, con un diseño de encabezado diferente (por ejemplo, puertos de 32 bits), NAT fallará. Toda la idea de los protocolos de capas es asegurar que los cambios en una capa no requieran cambios en otras capas. NAT destruye esta independencia.

Cuarto, en Internet no se exige que los procesos utilicen TCP o UDP. Si un usuario en la máquina *A* decide usar algún nuevo protocolo de transporte para hablar con un usuario en la máquina *B* (por ejemplo, para una aplicación multimedia), la introducción de una caja NAT hará que la aplicación falle porque la caja NAT no podrá localizar el *Puerto de origen* TCP correctamente.

Quinto, algunas aplicaciones insertan direcciones IP en el cuerpo del texto. El receptor extrae estas direcciones y las usa. Puesto que NAT no sabe nada sobre estas direcciones, no las puede reemplazar, de modo que fallará cualquier intento de usarlas en el extremo remoto. El **FTP** (**Protocolo de Transferencia de Archivos**) estándar funciona de esta manera y puede fallar en presencia del NAT a menos que se tomen precauciones especiales. Del mismo modo, el protocolo de telefonía H.323 de Internet (que estudiaremos en el capítulo 7) tiene esta propiedad y puede fallar en presencia del NAT. Puede ser posible arreglar NAT para que trabaje con H.323, pero no es una buena idea tener que arreglar el código en la caja NAT cada vez que surge una nueva aplicación.

Sexto, debido a que el campo *Puerto de origen* de TCP es de 16 bits, a lo sumo se pueden asignar 65,536 máquinas hacia una dirección IP. En realidad, el número es ligeramente menor porque los primeros 4096 puertos se reservan para usos especiales. Sin embargo, si hay varias direcciones IP disponibles, cada una puede manejar 61,440 máquinas.

En el RFC 2993 se explican éstos y otros problemas con NAT. En general, los antagonistas de NAT dicen que arreglando el problema de las direcciones IP insuficientes de esta manera temporal y poco elegante, la presión de implementar la solución real, es decir la transición a IPv6, se reduce, y el problema persiste.

5.6.3 Protocolos de Control en Internet

Además del IP que se usa para transferencia de datos, Internet tiene algunos protocolos de control que se usan en la capa de redes, como ICMP, ARP, RARP, BOOTP y DHCP. En esta sección veremos cada uno a su vez.

Protocolo de Mensajes de Control en Internet

Los enrutadores supervisan estrechamente el funcionamiento de Internet. Cuando ocurre algo inesperado, el **Protocolo de Mensajes de Control en Internet** (**ICMP**) informa del evento, que también se utiliza para probar Internet. Hay definidos alrededor de una docena de tipos de mensajes ICMP. Los más importantes se listan en la figura 5-61. Cada tipo de mensaje ICMP se encapsula en un paquete IP.

Tipo de mensaje	Descripción
Destination unreachable	El paquete no se pudo entregar
Time exceeded	Campo de tiempo de vida = 0
Parameter problem	Campo de encabezado no válido
Source quench	Paquete regulador
Redirect	Enseña a un enrutador sobre geografía
Echo	Pregunta a una máquina si está viva
Echo reply	Sí, estoy viva
Timestamp request	Misma que solicitud de eco, pero con marca de tiempo
Timestamp reply	Misma que respuesta de eco, pero con marca de tiempo

Figura 5-61. Los principales tipos de mensaje ICMP.

El mensaje DESTINATION UNREACHABLE se usa cuando la subred o un enrutador no pueden localizar el destino o cuando un paquete con el bit *DF* no puede entregarse porque una red de "paquete pequeño" se posiciona en la ruta.

El mensaje TIME EXCEEDED se envía cuando un paquete se cae porque su contador ha llegado a cero. Este evento es un síntoma de que los paquetes se están repitiendo, que hay una congestión enorme, o que los valores del cronómetro se han fijado demasiado bajos.

El mensaje PARAMETER PROBLEM indica que se ha descubierto un valor ilegal en un campo de encabezado. Este problema indica un error en el software de IP del *host* que envía o posiblemente en el software de un enrutador de tránsito.

El mensaje SOURCE QUENCH se utilizaba anteriormente para regular a los *hosts* que estaban enviando demasiados paquetes. Se esperaba que cuando un *host* recibiera este mensaje redujera la velocidad. Se usa cada vez menos porque cuando ocurre la congestión, estos paquetes tienden a agravar más la situación. Ahora el control de congestión en Internet se hace sobre todo en la capa de transporte; lo estudiaremos en detalle en el capítulo 6.

El mensaje REDIRECT se usa cuando un enrutador se percata de que un paquete parece estar mal enrutado. Lo utiliza el enrutador para avisar al *host* emisor sobre el probable error.

Los mensajes ECHO y ECHO REPLY se utilizan para ver si un destino dado es alcanzable y está vivo. Se espera que el destino envíe de vuelta un mensaje ECHO REPLY luego de recibir el mensaje ECHO. Los mensajes TIMESTAMP REQUEST y TIMESTAMP REPLY son similares, sólo que el tiempo de la llegada del mensaje y la salida de la contestación se graban en la contestación. Esta característica se usa para medir el rendimiento de la red.

Además de estos mensajes, se han definido otros. La lista en línea se conserva ahora en *www.iana.org/assignments/icmp-parameters.*

ARP—Protocolo de Resolución de Direcciones

Aunque en Internet cada máquina tiene una (o más) direcciones IP, en realidad éstas no pueden usarse para enviar paquetes porque el hardware de capa de enlace de datos no entiende las direcciones de Internet. Hoy día, la mayoría de los *hosts* en las compañías y universidades se une a una LAN por una tarjeta de red que sólo entiende direcciones LAN. Por ejemplo, cada tarjeta Ethernet viene provista de fábrica con una dirección Ethernet de 48 bits. Los fabricantes de tarjetas Ethernet solicitan un bloque de direcciones a una autoridad central para asegurar que dos tarjetas no tengan la misma dirección (para evitar los conflictos de si las dos tarjetas deban aparecer en la misma LAN). Las tarjetas envían y reciben tramas basadas en direcciones Ethernet de 48 bits. No saben nada de direcciones IP de 32 bits.

La pregunta ahora es: ¿cómo se convierten direcciones IP en direcciones de capa de enlace de datos, como Ethernet? Para explicar cómo funciona esto, veamos el ejemplo de la figura 5-62 en que se ilustra una universidad pequeña con algunas redes de clase C (ahora llamadas /24). Aquí tenemos dos Ethernets, una en el Depto. de Informática, con dirección IP 192.31.65.0 y otra en Ingeniería Eléctrica, con dirección IP 192.31.63.0. Éstas están conectadas por un anillo de red dorsal del campus (por ejemplo, FDDI) con la dirección IP 192.31.60.0. Cada máquina en una Ethernet tiene una dirección única de Ethernet, etiquetadas de *E1* a *E6*, y cada máquina en el anillo de FDDI tiene una dirección de FDDI, etiquetada de *F1* a *F3*.

Empezaremos viendo cómo un usuario en el *host* 1 envía un paquete a un usuario en el *host* 2. Supongamos que el emisor sabe el nombre del receptor pretendido, posiblemente algo como *mary@eagle.cs.uni.edu.* El primer paso es encontrar la dirección IP para el *host* 2, conocido como *eagle.cs.uni.edu.* Esta consulta la realiza el Sistema de Nombres de Dominio que estudiaremos en el capítulo 7. De momento, asumiremos que DNS devuelve la dirección IP al *host* 2 (192.31.65.5).

El software de la capa superior en el *host* 1 elabora ahora un paquete con 192.31.65.5 en el campo *Dirección de destino* y lo da a transmitir al software IP. Éste puede buscar la dirección y ver que el destino esté en su propia red, pero necesita alguna manera de encontrar la dirección Ethernet de destino. Una solución es tener un archivo de configuración en alguna parte del sistema

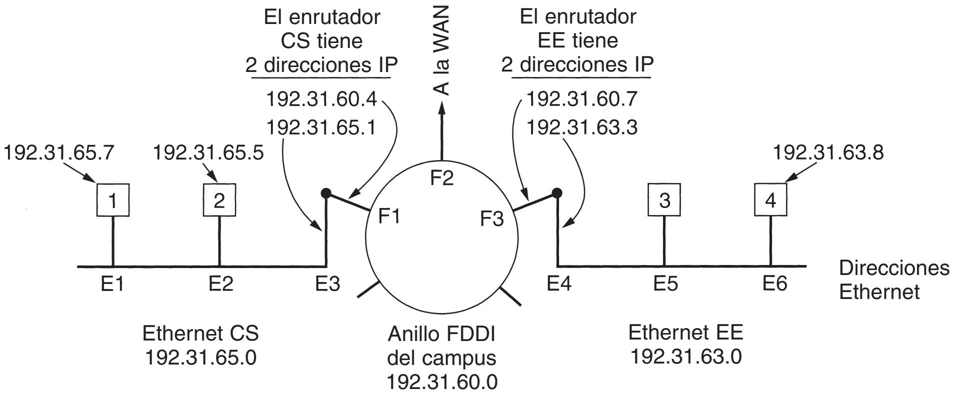


Figura 5-62. Tres redes /24 interconectadas: dos Ethernets y un anillo FDDI.

que relacione direcciones IP con direcciones Ethernet. Aun cuando esta solución es ciertamente posible, para las organizaciones con miles de máquinas, conservar todos estos archivos actualizados propicia errores y consume mucho tiempo.

Una mejor solución es que el *host* 1 dé salida a un paquete de difusión hacia Ethernet preguntando: ¿quién posee la dirección IP 192.31.65.5? La difusión llegará a cada máquina en Ethernet 192.31.65.0, y cada una verificará su dirección IP. Al *host* 2 le bastará responder con su dirección de Ethernet (*E2*). De esta manera, el *host* 1 aprende que esa dirección IP 192.31.65.5 está en el *host* con la dirección Ethernet *E2*. El protocolo utilizado para hacer esta pregunta y obtener la respuesta se llama **Protocolo de Resolución de Direcciones** (**ARP**)**.** Casi cada máquina en Internet lo ejecuta. La definición de ARP está en el RFC 826.

La ventaja de usar ARP en lugar de archivos de configuración es la sencillez. El gerente de sistemas sólo tiene que asignar a cada máquina una dirección IP y decidir respecto de las máscaras de subred. ARP hace el resto.

A estas alturas, el software IP en el *host* 1 crea una trama Ethernet dirigida a *E2,* pone el paquete IP (dirigido a 192.31.65.5) en el campo de carga útil, y lo descarga hacia la Ethernet. La tarjeta Ethernet del *host* 2 detecta esta trama, la reconoce como una trama para sí mismo, lo recoge, y provoca una interrupción. El controlador de Ethernet extrae el paquete IP de la carga útil y lo pasa al software IP, que ve que esté direccionado correctamente y lo procesa.

Se pueden hacer varias optimizaciones para que ARP trabaje con más eficiencia. Para empezar, una vez que una máquina ha ejecutado ARP, guarda el resultado en caso de tener que ponerse en poco tiempo de nuevo en contacto con la misma máquina. La siguiente vez encontrará la correspondencia en su propio caché, eliminando así la necesidad de una segunda difusión. En muchos casos el *host* 2 necesitará devolver una respuesta, forzando, también, a que se ejecute el ARP para determinar la dirección Ethernet del emisor. Esta difusión de ARP puede evitarse teniendo el *host* 1

que incluir su correspondencia IP a Ethernet en el paquete ARP. Cuando la difusión de ARP llega al *host* 2, se introduce (192.31.65.7, E1) en el caché ARP del *host* 2 para uso futuro. De hecho, todas las máquinas en Ethernet pueden introducir esta correspondencia en su caché ARP.

Otra optimización más es que cada máquina difunda su correspondencia cuando arranca. Por lo general, esta difusión se hace en forma de un ARP que busca su propia dirección IP. No debe haber una respuesta, pero un efecto lateral de la difusión es hacer una entrada en el caché ARP de todas las máquinas. Si llega (inesperadamente) una respuesta, es que la misma dirección IP se ha asignado a dos máquinas. La más reciente debe informar al gerente de sistemas y no arrancar.

Para permitir que las correspondencias cambien, por ejemplo, cuando una tarjeta Ethernet falla y se reemplaza con una nueva (y, por lo tanto, una nueva dirección Ethernet), las entradas en el caché ARP deben expirar en unos cuantos minutos.

Ahora veamos de nuevo la figura 5-62. Esta vez el *host* 1 quiere enviar un paquete al *host* 4 (192.31.63.8). Si utiliza ARP fallará porque el *host* 4 no verá la difusión (los enrutadores no envían difusiones a nivel Ethernet). Hay dos soluciones. Primera, podría configurarse el enrutador CS para que responda a las solicitudes de ARP de la red 192.31.63.0 (y posiblemente del otras redes locales). En este caso, el *host* 1 introducirá (192.31.63.8, E3) en el caché ARP y enviará felizmente todo el tráfico del *host* 4 al enrutador local. Esta solución se llama **proxy ARP.** La segunda solución es hacer que el *host* 1 vea de inmediato que el destino está en una red remota y simplemente envíe todo ese tráfico a una dirección Ethernet predefinida que se ocupe de todo el tráfico remoto, en este caso *E3.* Esta solución no requiere que el enrutador CS sepa a qué redes remotas está sirviendo.

De cualquier modo, lo que sucede es que el *host* 1 empaca el paquete IP en el campo de carga útil de una trama Ethernet dirigida a *E3*. Cuando el enrutador CS obtiene la trama Ethernet, retira el paquete IP del campo de carga útil y busca la dirección IP en sus tablas de enrutamiento. Descubre que se supone que los paquetes para la red 192.31.63.0 van al enrutador 192.31.60.7. Si aún no conoce la dirección FDDI de 192.31.60.7, transmite un paquete ARP al anillo y aprende que su dirección del anillo es *F3.* Inserta entonces el paquete en el campo de carga útil de una trama FDDI dirigido a *F3* y la coloca en el anillo.

En el enrutador EE, el controlador de FDDI retira el paquete del campo de carga útil y lo da al software IP, que ve que necesita enviar el paquete a 192.31.63.8. Si esta dirección IP no está en su caché ARP, transmite una solicitud de ARP en la Ethernet EE y aprende que la dirección de destino es *E6,* por lo que construye una trama Ethernet dirigida a *E6*, pone el paquete en el campo de carga útil y lo envía a través de Ethernet. Cuando la trama Ethernet llega al *host* 4, se extrae el paquete de la trama y se pasa al software IP para su procesamiento.

Ir del *host* 1 a una red distante a través de una WAN funciona esencialmente de la misma manera, sólo que esta vez las tablas del enrutador CS le dicen que utilice el enrutador de WAN cuya dirección FDDI es *F2.*

RARP, BOOTP y DHCP

ARP resuelve el problema de encontrar qué dirección Ethernet corresponde a una dirección IP dada. A veces se tiene que resolver el problema inverso: dada una dirección Ethernet, ¿cuál es la dirección IP correspondiente? En particular, este problema ocurre cuando se inicializa una estación de trabajo sin disco. Dicha máquina recibirá normalmente la imagen binaria de su sistema operativo desde un servidor de archivos remoto. ¿Pero cómo aprende su dirección IP?

La primera solución inventada fue usar el **RARP** (**Protocolo de Resolución de Dirección de Retorno**) (su definición está en el RFC 903). Este protocolo permite que una estación de trabajo recientemente inicializada transmita su dirección Ethernet y diga: "Mi dirección Ethernet de 48 bits es 14.04.05.18.01.25. ¿Alguien allá afuera conoce mi dirección IP?" El servidor RARP ve esta solicitud, busca la dirección Ethernet en sus archivos de configuración y devuelve la dirección IP correspondiente.

Usar RARP es mejor que incrustar una dirección IP en la imagen de memoria porque esto permite usar la misma imagen en todas las máquinas. Si la dirección IP se incrustara en la imagen, cada estación de trabajo necesitaría su propia imagen.

Una desventaja de RARP es que usa una dirección de destino de todos los 1s (de difusión limitada) para llegar al servidor RARP. Sin embargo, dichas difusiones no las envían los enrutadores, por lo que se necesita un servidor RARP en cada red. Para resolver este problema, se inventó un protocolo de arranque alternativo llamado **BOOTP.** A diferencia de RARP, el BOOTP usa mensajes UDP que se envían a través de los enrutadores. También proporciona información adicional a una estación de trabajo sin disco, incluso la dirección IP del servidor de archivos que contiene la imagen de memoria, la dirección IP del enrutador predeterminado y la máscara de subred que debe usar. BOOTP se describe en los RFCs 951, 1048 y 1084.

Un problema serio con BOOTP es que requiere configuración manual de tablas para relacionar una dirección IP con una dirección Ethernet. Cuando se agrega un nuevo *host* a una LAN, no se puede usar el BOOTP hasta que un administrador le haya asignado una dirección IP e introducido manualmente sus direcciones IP y Ethernet en las tablas de configuración de BOOTP. Para eliminar este paso conducente a errores, el BOOTP se extendió y se le dio un nuevo nombre: **DHCP** (**Protocolo de Configuración de *Host* Dinámico**)**.** DHCP permite asignación de dirección IP manual y automática. Se describe en los RFCs 2131 y 2132. En la mayoría de los sistemas, ha reemplazado a RARP y BOOTP.

Como RARP y BOOTP, DHCP se basa en la idea de un servidor especial que asigna direcciones IP a *hosts* que las requieren. Este servidor no necesita estar en la misma LAN que el *host* solicitante. Puesto que el servidor DHCP no se puede alcanzar por difusión, se necesita un **agente de retransmisión DHCP** en cada LAN, como se muestra en la figura 5-63.

Para encontrar su dirección IP, una máquina inicializada recientemente difunde un paquete DHCP DISCOVER. El agente de retransmisión DHCP de su LAN intercepta todas las difusiones DHCP. Cuando encuentra un paquete DHCP DISCOVER, envía el paquete mediante unidifusión al servidor

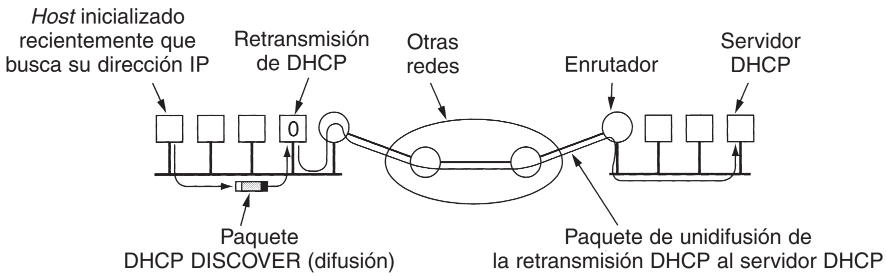


Figura 5-63. Funcionamiento de DHCP.

DHCP, posiblemente en una red distante. La única pieza de información que el agente de retransmisión necesita es la dirección IP del servidor DHCP.

Un problema que surge con la asignación automática de direcciones IP de un rango de direcciones, es por cuánto tiempo debe asignarse una dirección IP. Si un *host* deja la red y no devuelve su dirección IP al servidor DHCP, esa dirección se perderá permanentemente. Después de un periodo, pueden perderse muchas direcciones. Para impedir que eso pase, la asignación de dirección IP puede ser por un periodo fijo, una técnica llamada **arrendamiento.** Simplemente, antes de que expire el arriendo, el *host* debe pedirle una renovación al DHCP. Si no hace una solicitud o ésta se le niega, el *host* ya no puede usar la dirección IP que se le dio antes.

5.6.4 OSPF—Protocolos de enrutamiento de puerta de enlace interior

Hemos terminado nuestro estudio de protocolos de control de Internet. Es tiempo para pasar al tema siguiente: el enrutamiento en Internet. Como lo mencionamos antes, Internet se compone de una gran cantidad de sistemas autónomos. Cada uno de ellos es manejado por una organización diferente y puede usar su propio algoritmo interno de enrutamiento. Por ejemplo, las redes internas de las compañías *X, Y* y *Z* que normalmente se ven como tres sistemas autónomos si los tres están en Internet. Los tres pueden usar internamente algoritmos de enrutamiento diferentes. No obstante, siguiendo los estándares incluso para enrutamiento interno, simplifica la implementación de los límites entre los sistemas autónomos y permite reutilizar el código. En esta sección estudiaremos el enrutamiento dentro de un sistema autónomo. En la siguiente veremos el enrutamiento entre sistemas autónomos. Un algoritmo de enrutamiento dentro de un sistema autónomo se llama **protocolo de puerta de enlace interior (IGP);** un algoritmo para enrutamiento entre sistemas autónomos se llama **protocolo de puerta de enlace exterior (EGP).**

El protocolo de puerta de enlace interior original de Internet era un protocolo de vector de distancia (RIP) basado en el algoritmo de Bellman-Ford heredado de ARPANET. Funcionó bien en sistemas pequeños, pero no así conforme los sistemas autónomos fueron más grandes. También padeció el problema de la cuenta hasta el infinito y la convergencia generalmente lenta, por lo que se reemplazó en mayo de 1979 por un protocolo de estado del enlace. En 1988, la Fuerza de Tarea de Ingeniería de Internet empezó el trabajo en un sucesor. Ese sucesor, llamado **OSPF (Abrir**

Primero la Ruta más Corta), se volvió una norma en 1990. Ahora la mayoría de vendedores de enrutadores lo apoyan, y se ha convertido en el protocolo de puerta de enlace interior principal. A continuación daremos un bosquejo de cómo funciona OSPF. Para la historia completa, vea el RFC 2328.

Dada la larga experiencia con otros protocolos de enrutamiento, el grupo que diseñó el nuevo protocolo tenía una larga lista de requisitos que cumplir. Primero, el algoritmo se tenía que publicar en la literatura abierta, de ahí la "O" inicial de OSPF. Una solución patentada poseída por una compañía no lo haría. Segundo, el nuevo protocolo tenía que apoyar una variedad de métrica de distancia, como la distancia física, retardo, etcétera. Tercero, tenía que ser un algoritmo dinámico, uno que se adaptara automática y rápidamente a los cambios de topología.

Cuarto, y esto era nuevo para OSPF, tenía que apoyar el enrutamiento con base en el tipo de servicio. El nuevo protocolo tenía que poder dirigir el tráfico en tiempo real de una manera y el resto del tráfico de una manera diferente. El protocolo IP tiene un campo *Tipo de servicio,* pero ningún protocolo de enrutamiento existente lo usó. Este campo estaba incluido en OSPF pero tampoco se usó, y finalmente lo eliminaron.

Quinto, y relacionado con el anterior, el nuevo protocolo tenía que balancear la carga, dividiéndola en líneas múltiples. La mayoría de los protocolos anteriores enviaba todos los paquetes por la mejor ruta. La mejor segunda ruta no se usó en absoluto. En muchos casos, dividir la carga en líneas múltiples ofrece un mejor desempeño.

Sexto, se necesitó apoyo para los sistemas jerárquicos. En 1988 Internet había crecido tanto que no se podía esperar que ningún enrutador conociera toda la topología. Se tuvo que diseñar el nuevo protocolo de enrutamiento para que ningún enrutador tuviera que conocerla.

Séptimo, se requirió una pizca de seguridad para impedir que los estudiantes bromistas engañaran a los enrutadores enviándoles falsa información de enrutamiento. Finalmente, se necesitó una previsión para tratar con enrutadores que se conectaban a Internet mediante un túnel. Los protocolos anteriores no manejaron bien este aspecto.

OSPF soporta tres tipos de conexiones y redes:

1. Las líneas punto a punto exactamente entre dos enrutadores.
2. Redes de multiacceso con difusión (por ejemplo, la mayoría de las LANs).
3. Redes de multiacceso sin difusión (por ejemplo, la mayoría de las WANs de paquetes conmutados).

Una red de **multiacceso** es la que puede tener múltiples enrutadores, cada uno de los cuales se puede comunicar directamente con todos los demás. Todas las LANs y WANs tienen esta propiedad. La figura 5-64(a) muestra un sistema autónomo que contiene los tres tipos de redes. Observe que en general los *hosts* no tienen un papel en OSPF.

OSPF funciona resumiendo la colección de redes reales, enrutadores y líneas en un grafo dirigido en el que a cada arco se asigna un costo (distancia, retardo, etcétera). Entonces calcula la ruta más corta con base en los pesos de los arcos. Una conexión en serie entre dos enrutadores se representa por un par de arcos, uno en cada dirección. Sus pesos pueden ser diferentes. Una red de

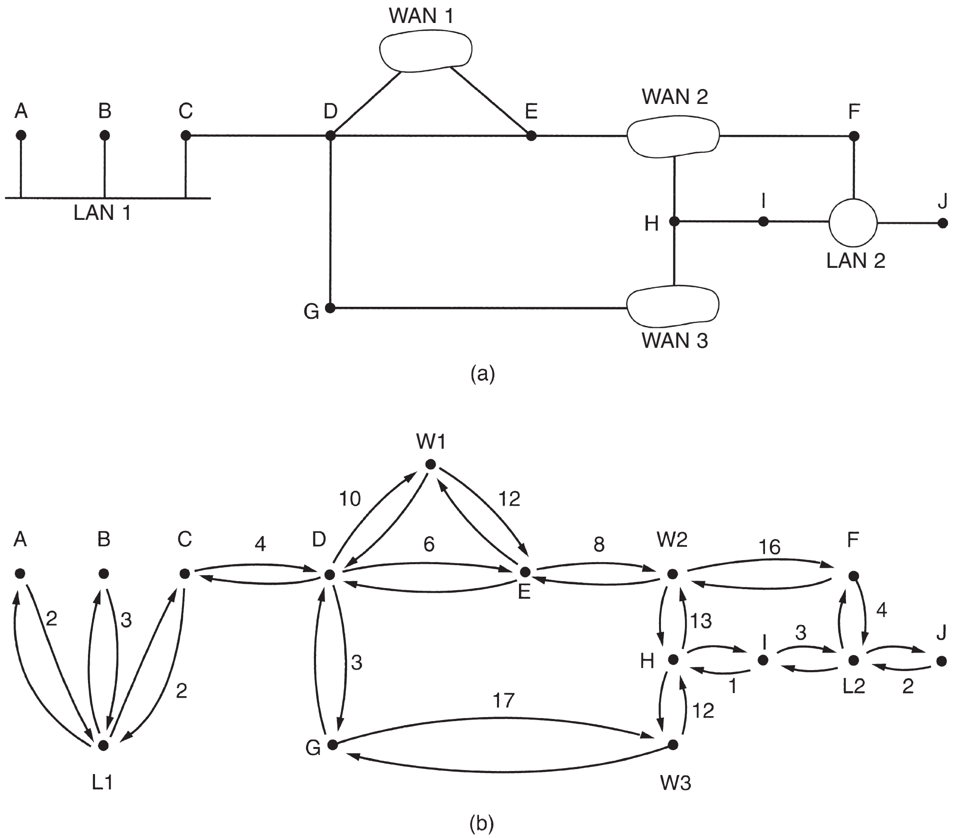


Figura 5-64. (a) Un sistema autónomo. (b) Representación gráfica de (a).

multiacceso se representa con un nodo para la red en sí más un nodo para cada enrutador. Los arcos del nodo de la red a los enrutadores tienen peso 0 y se omiten del grafo.

La figura 5-64(b) muestra la representación gráfica de la red de la figura 5-64(a). Los pesos son simétricos, a menos que se marcaran de otra manera. Lo que OSPF hace fundamentalmente es representar la red real como un grafo y entonces calcular el camino más corto de uno a otro enrutador.

Muchos de los sistemas autónomos en Internet son grandes por sí mismos y nada sencillos de administrar. OSPF les permite dividirlos en **áreas** numeradas donde un área es una red o un conjunto de redes inmediatas. Las áreas no se traslapan ni la necesidad es exhaustiva, es decir, algunos enrutadores no pueden pertenecer a área alguna. Un área es una generalización de una subred. Fuera de un área, su topología y detalles no son visibles.

Cada sistema autónomo tiene un área de **red dorsal,** llamada 0. Todas las áreas se conectan a la red dorsal, posiblemente por túneles, de modo que es posible entrar desde cualquier área en el sistema autónomo a cualquier otra área en el sistema autónomo mediante la red dorsal. En el grafo un túnel se representa como un arco y tiene un costo. Cada enrutador que se conecta a dos o

más áreas es parte de la red dorsal. Como con otras áreas, la topología de la red dorsal no es visible fuera de ésta.

Dentro de un área, cada enrutador tiene la misma base de datos del estado del enlace y ejecuta el mismo algoritmo de la ruta más corta. Su trabajo principal es calcular el camino más corto desde sí mismo a cualquier otro enrutador en el área, incluso el enrutador que se conecta a la red dorsal, de la que debe haber una por lo menos. Un enrutador que conecta dos áreas necesita las bases de datos para las dos áreas y debe ejecutar el algoritmo de la ruta más corta por separado.

Durante la operación normal, pueden necesitarse tres tipos de rutas: dentro del área, entre áreas y entre sistemas autónomos. Las rutas dentro del área son las más fáciles, puesto que el enrutador de origen ya conoce el camino más corto al enrutador de destino. El enrutamiento entre áreas siempre procede en tres pasos: va del origen a la red dorsal; va a través de la red dorsal al área de destino; va al destino. Este algoritmo fuerza una configuración de estrella en OSPF con la red dorsal actuando como concentrador y las otras áreas como rayos. Los paquetes se enrutan del origen al destino “como están”. No se encapsulan ni se entunelan, a menos que vayan a un área cuya única conexión a la red dorsal sea un túnel. La figura 5-65 muestra parte de Internet con sistemas autónomos y áreas.

OSPF distingue cuatro clases de enrutadores:

1. Enrutadores internos que están totalmente dentro de un área.
2. Enrutadores de límite de área que conectan dos o más áreas.
3. Enrutadores de la red dorsal que están en la red dorsal.
4. Enrutadores fronterizos de sistemas autónomos que se comunican con los enrutadores de otros sistemas autónomos.

Estas clases se pueden traslapar. Por ejemplo, todos los enrutadores de límite de área forman parte de la red dorsal automáticamente. Además, un enrutador que está en la red dorsal pero no forma parte de cualquier otra área también es un enrutador interno. En la figura 5-65 se ilustran ejemplos de las cuatro clases de enrutadores.

Cuando un enrutador se inicia, envía mensajes HELLO en todas sus líneas punto a punto y los multidifunde en las LANs al grupo que contiene los enrutadores restantes. En las WANs, necesita alguna información de configuración para saber a quién contactar. A partir de las respuestas, cada enrutador aprende quiénes son sus vecinos. Todos los enrutadores en la misma LAN son vecinos.

OSPF trabaja intercambiando información entre enrutadores adyacentes, que no es lo mismo que entre enrutadores vecinos. En particular, es ineficaz tener cualquier enrutador en la LAN que se comunica con cualquier otro enrutador en la LAN. Para evitar esta situación, se elige un enrutador como **enrutador designado.** Se dice que es **adyacente** a todos los demás enrutadores en su LAN, e intercambia información con ellos. Los enrutadores vecinos que no son adyacentes no intercambian información entre sí. Un enrutador designado como respaldo siempre se guarda actualizado, para facilitar la transición en caso de que el primer enrutador designado se cayera y necesitara ser reemplazado de manera inmediata.

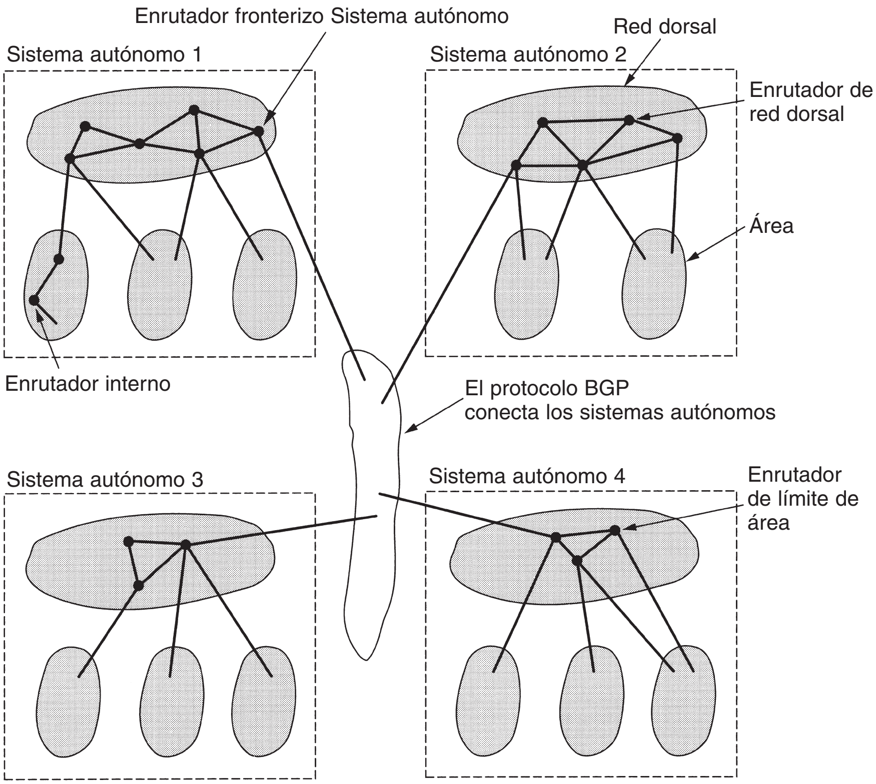


Figura 5-65. Relación entre sistemas autónomos, redes dorsales y áreas en OSPF.

Durante la operación normal, cada enrutador inunda periódicamente con mensajes LINK STATE UPDATE a cada uno de sus enrutadores adyacentes. Este mensaje da su estado y proporciona los costos usados en la base de datos topológica. Para hacerlos confiables, se confirma la recepción de los mensajes de inundación. Cada mensaje tiene un número de secuencia para que un enrutador pueda ver si un LINK STATE UPDATE entrante es más viejo o más nuevo que el que tiene actualmente. Los enrutadores también envían estos mensajes cuando una línea sube o baja o su costo cambia.

Los mensajes DATABASE DESCRIPTION dan los números de secuencia de todas las entradas de estado del enlace poseídas por el emisor actualmente. Comparando sus propios valores con los del emisor, el receptor puede determinar quién tiene los valores más recientes. Estos mensajes se usan cuando se activa una línea.

Cualquier socio puede pedir información del estado del enlace al otro usando los mensajes LINK STATE REQUEST. El resultado de este algoritmo es que cada par de enrutadores adyacentes hace una verificación para ver quién tiene los datos más recientes, y de esta manera se difunde la nueva información a lo largo del área. Todos estos mensajes se envían como paquetes IP. En la figura 5-66 se resumen los cinco tipos de mensajes.

Tipo de mensaje	Descripción
Hello	Descubre quiénes son los vecinos
Link state update	Proporciona los costos del emisor a sus vecinos
Link state ack	Confirma la recepción de la actualización del estado del enlace
Database description	Anuncia qué actualizaciones tiene el emisor
Link state request	Solicita información del socio

Figura 5-66. Los cinco tipos de mensajes de OSPF.

Finalmente podemos reunir todas las piezas. Utilizando la inundación de mensajes, cada enrutador informa a todos los demás enrutadores en su área sobre sus vecinos y costos. Esta información permite a cada enrutador construir el grafo para su(s) área(s) y calcular la ruta más corta. El área de la red dorsal también hace esto. Además, los enrutadores de la red dorsal aceptan la información de los enrutadores del límite de área para calcular la mejor ruta de cada enrutador de la red dorsal a cada enrutador restante. Esta información se difunde a los enrutadores de límite de área que la anuncian dentro de sus áreas. Usando esta información, un enrutador que está a punto de enviar un paquete dentro del área puede seleccionar el enrutador de mejor salida a la red dorsal.

5.6.5 BGP—Protocolo de Puerta de Enlace de Frontera

Dentro de un solo sistema autónomo, el protocolo de enrutamiento recomendado es OSPF (aunque no es ciertamente el único en uso). Entre los sistemas autónomos se utiliza un protocolo diferente, el **Protocolo de Puerta de Enlace de Frontera** (**BGP**). Se necesita un protocolo diferente entre sistemas autónomos porque los objetivos de un protocolo de puerta de enlace interior y un protocolo de puerta de enlace exterior no son los mismos. Todo lo que tiene que hacer un protocolo de puerta de enlace interior es mover lo más eficazmente posible los paquetes del origen al destino. No tiene que preocuparse por las políticas.

Los enrutadores del protocolo de puerta de enlace exterior tienen que preocuparse en gran manera por la política (Metz, 2001). Por ejemplo, un sistema autónomo corporativo podría desear la habilidad para enviar paquetes a cualquier sitio de Internet y recibir los paquetes de cualquier sitio de Internet. Sin embargo, podría no estar dispuesto a llevar paquetes de tránsito que se originan en un sistema autónomo foráneo con destino a un sistema autónomo foráneo diferente, aun cuando su propio sistema autónomo estaba en la ruta más corta entre los dos sistemas autónomos foráneos ("Ése es su problema, no el nuestro"). Por otro lado, podría estar dispuesto a llevar el tráfico del tránsito para sus vecinos o incluso para otros sistemas autónomos específicos que pagaron por este servicio. Por ejemplo, las compañías de teléfonos podrían estar contentas de actuar como empresas portadoras para sus clientes, pero no para otros. En general, los protocolos de puerta de enlace exterior, y BGP en particular, se han diseñado para permitir que se implementen muchos tipos de políticas de enrutamiento en el tráfico entre sistemas autónomos.

Las políticas típicas implican consideraciones políticas, de seguridad, o económicas. Algunos ejemplos de limitaciones de enrutamiento son:

1. Ningún tránsito a través de ciertos sistemas autónomos.
2. Nunca ponga Irak en una ruta que inicie en el Pentágono.
3. No pasar por Estados Unidos para llegar de la Columbia Británica a Ontario.
4. Transite por Albania sólo si no hay otra alternativa al destino.
5. El tráfico que empieza o termina en IBM no debe transitar por Microsoft.

Las políticas en cada enrutador de BGP se configuran manualmente (o usando algún tipo de escritura). No son parte del protocolo.

Desde el punto de vista de un enrutador de BGP, el mundo consiste en sistemas autónomos y las líneas que los conectan. Dos sistemas autónomos se consideran conectados si hay una línea entre un enrutador fronterizo en cada uno. Dado el especial interés de BGP en el transporte de tráfico, las redes se agrupan en una de tres categorías. La primera son las **redes *stub*,** que tienen sólo una conexión con el grafo de BGP. Éstas no se pueden usar para transportar tráfico porque no hay nadie en el otro lado. Luego vienen las **redes multiconectadas.** Éstas podrían usarse para el transporte de tráfico excepto que lo rechacen. Finalmente, están las **redes de tránsito,** como redes dorsales, que están dispuestas a ocuparse de paquetes de terceros, posiblemente con algunas restricciones, y normalmente por pago.

Los pares de enrutadores de BGP se comunican entre sí estableciendo conexiones TCP. Operando de esta manera proporcionan comunicación confiable y ocultan todo detalle de red que pase a través de ellos.

Básicamente, BGP es muy parecido a un protocolo de vector de distancia, pero muy diferente de la mayoría de otros como RIP. En lugar de mantener el costo para cada destino, cada enrutador de BGP guarda el registro de la ruta utilizada, por lo que se conoce como un protocolo de vector de ruta. Del mismo modo, en lugar de darle a cada vecino el costo de cada posible destino estimado periódicamente, cada enrutador de BGP les dice el camino exacto que está usando.

Por ejemplo, considere los enrutadores de BGP mostrados en figura 5-67(a). En particular, considere la tabla de enrutamiento de *F*. Suponga que utiliza la ruta *FGCD* para llegar a *D*. Cuando los vecinos le dan información de la ruta, le proporcionan sus rutas completas, como se muestra en la figura 5-67(b) (para simplificar, sólo se muestra aquí el destino *D*).

Luego que han llegado todas las rutas de los vecinos, *F* las examina para ver cuál es la mejor. Desecha pronto las de *I* y *E*, porque pasan a través de la propia *F*. La opción está entonces entre usar *B* y *G*. Cada enrutador de BGP contiene un módulo que examina las rutas a un destino dado y las califica, devolviendo un número para la "distancia" a ese destino por cada ruta. Cualquier ruta que viole una restricción de la política recibe automáticamente una calificación al infinito. Entonces el enrutador adopta la ruta de la distancia más corta. La función de calificar no es parte del protocolo de BGP y puede ser cualquier función que el gerente de sistemas desee.

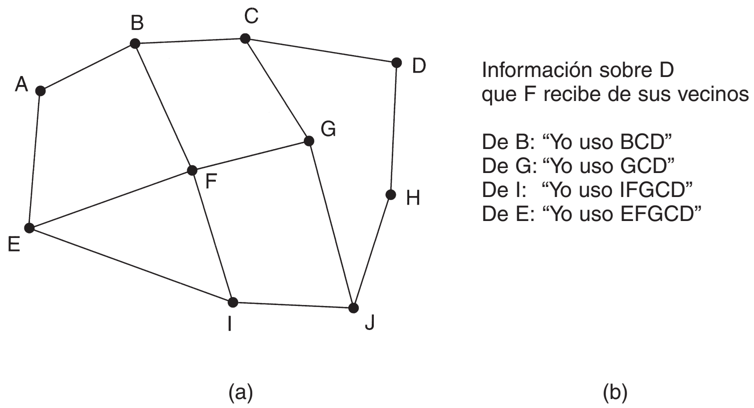


Figura 5-67. (a) Conjunto de enrutadores de BGP. (b) Información enviada a F.

BGP resuelve fácilmente el problema de la cuenta hasta el infinito que plaga otros algoritmos de vector de distancia. Por ejemplo, suponga que *G* se congela o que la línea *FG* se cae. Entonces *F* recibe las rutas de sus tres vecinos restantes. Estas rutas son *BCD, IFGCD* y *EFGCD*. Puede ver inmediatamente que las dos últimas rutas son vanas, ya que atraviesan *F*, por lo que escoge *FBCD* como su nueva ruta. A menudo otros algoritmos de vector de distancia hacen una mala elección porque no saben quiénes de sus vecinos tienen rutas independientes al destino y quiénes no. La definición de BGP está en los RFCs 1771 a 1774.

5.6.6 Multidifusión de Internet

La comunicación normal de IP está entre un emisor y un receptor. Sin embargo, para algunas aplicaciones es útil que un proceso pueda enviar simultáneamente a una gran cantidad de receptores, por ejemplo, actualización duplicada, bases de datos distribuidas, transmisión de cotizaciones de acciones a corredores múltiples, y manejo de conferencia digital en llamadas telefónicas (es decir, entre muchas partes).

IP apoya la multidifusión, usando direcciones clase D. Cada dirección clase D identifica un grupo de *hosts*. Hay 28 bits disponibles para identificar los grupos, de modo que pueden existir al mismo tiempo más de 250 millones de grupos. Cuando un proceso envía un paquete a una dirección clase D, se hace el mejor esfuerzo para entregarlo a todos los miembros del grupo direccionado, pero no se da garantía alguna. Quizá algunos miembros no reciban el paquete.

Se soportan dos tipos de direcciones de grupo: las permanentes y las temporales. Un grupo permanente siempre está allí y no se tiene que preparar. Cada grupo permanente tiene una dirección de grupo permanente. Algunos ejemplos de direcciones de grupo permanentes son:

224.0.0.1 Todos los sistemas en una LAN
224.0.0.2 Todos los enrutadores en una LAN
224.0.0.5 Todos los enrutadores de OSPF en una LAN
224.0.0.6 Todos los enrutadores designados de OSPF en una LAN

Los grupos temporales se deben crear antes de que se puedan usar. Un proceso puede pedir a su *host* que se una a un grupo específico. También puede pedirle que deje el grupo. Cuando el último proceso en un *host* deja un grupo, ese grupo ya no está presente en el *host*. Cada *host* conserva el registro de qué grupos pertenecen actualmente a sus procesos.

La multidifusión se implementa mediante enrutadores de multidifusión especiales que pueden o no colocarse con los enrutadores normales. Alrededor de una vez por minuto, cada uno de estos enrutadores envía una multidifusión de hardware (es decir, una multidifusión de la capa de enlace de datos) a los *hosts* en su LAN (dirección 224.0.0.1) pidiéndoles que devuelvan información de los grupos a que pertenecen actualmente sus procesos. Cada *host* devuelve las respuestas a todas las direcciones clase D interesadas.

Estos paquetes de preguntas y respuestas utilizan un protocolo llamado **IGMP** (**Protocolo de Administración de Grupo de Internet**) que es vagamente análogo al ICMP. Tiene sólo dos tipos de paquetes: pregunta y respuesta, cada uno con un formato simple, fijo, que contiene alguna información de control en la primera palabra del campo de carga útil y una dirección clase D en la segunda palabra. Se describe en el RFC 1112.

El enrutamiento de multidifusión se crea utilizando árboles de difusión. Cada enrutador de multidifusión intercambia información con sus vecinos, usando un protocolo de vector de distancia modificado para que cada uno construya un árbol de expansión por grupo que cubra a todos los miembros del grupo. Se usan varias optimizaciones para recortar el árbol y eliminar los enrutadores y redes que no se interesan en grupos particulares. El protocolo hace un gran uso de túneles para no molestar a los nodos que no pertenecen al árbol de expansión.

5.6.7 IP móvil

Muchos usuarios de Internet tienen computadoras portátiles y desean permanecer conectados a Internet cuando visitan un sitio de Internet distante e incluso durante el camino. Desgraciadamente, el sistema de dirección IP facilita el trabajo lejos de casa más de dicho que de hecho. En esta sección examinaremos el problema y la solución. Una descripción más detallada se da en (Perkins, 1998a).

El villano real es el propio esquema de direccionamiento. Cada dirección IP contiene un número de red y un número de *host*. Por ejemplo, considere la máquina con dirección IP 160.80.40.20/16. El número de red es 160.80 (8272 en sistema decimal); 40.20 es el número de *host* (10260 en el decimal). Los enrutadores en todo el mundo tienen tablas de enrutamiento que indican qué línea usar para conseguir conectarse a la red 160.80. Siempre que un paquete llegue con un destino de dirección IP de la forma 160.80.xxx.yyy, saldrá de esa línea.

Si de repente la máquina con esa dirección se lleva fuera a algún sitio distante, los paquetes se le seguirán enrutando a su LAN principal (o enrutador). El dueño ya no podrá recibir más correo

electrónico, etcétera. Dar a la máquina una nueva dirección IP que corresponda a su nueva situación es poco atractivo porque se tendría que informar del cambio a una gran cantidad de personas, programas y bases de datos.

Otro método es que los enrutadores tengan que usar direcciones IP completas para enrutamiento, en lugar de sólo la red. Sin embargo, esta estrategia requeriría que cada enrutador tuviera millones de entradas de tablas, a un costo astronómico para Internet.

Cuando las personas empezaron a exigir la capacidad de conectar sus PCs portátiles a Internet dondequiera que estuvieran, la IETF preparó un grupo de trabajo para encontrar una solución. El grupo de trabajo formuló rápidamente varios objetivos considerados deseables para cualquier solución. Los principales fueron:

1. Cada *host* móvil debe poder usar su dirección IP principal en cualquier parte.
2. No se permiten cambios de software a los *hosts* fijos.
3. No se permiten cambios al software ni a las tablas del enrutador.
4. La mayoría de paquetes para *host* móviles no debe hacer desvíos en la ruta.
5. No se debe incurrir en sobrecarga cuando un *host* móvil está en casa.

La solución escogida fue la que se describe en la sección 5.2.8. Como un repaso breve, cada sitio que permita vagar a sus usuarios tiene que crear un agente de base. Cada sitio que permita visitantes tiene que crear un agente foráneo. Cuando un *host* móvil se presenta a un sitio foráneo, contacta al *host* foráneo y se registra. El *host* foráneo entonces contacta al agente de base del usuario y le da una **dirección temporal,** (*care*- of *address*) normalmente la propia dirección IP del agente foráneo.

Cuando un paquete llega a la LAN principal del usuario, entra a algún enrutador adjunto a la LAN. Por lo general, el enrutador trata de localizar al *host*, difundiendo un paquete ARP preguntando, por ejemplo: ¿cuál es la dirección Ethernet de 160.80.40.20? El agente de base responde a esta pregunta dando su propia dirección de Ethernet. El enrutador envía entonces los paquetes para 160.80.40.20 al agente principal. A su vez, este último los canaliza a la dirección temporal encapsulándolos en el campo de carga útil de un paquete IP dirigido al agente foráneo. El agente foráneo entonces lo desencapsula y lo entrega a la dirección del enlace de datos del *host* móvil. Además, el agente de base da la dirección temporal al emisor, para que los paquetes futuros se puedan canalizar directamente al agente foráneo. Esta solución reúne todos los requisitos declarados anteriormente.

Tal vez valga la pena mencionar un pequeño detalle. Cuando el *host* se mueve, el enrutador probablemente tenga su dirección Ethernet en el caché (próxima a quedar invalidada). Reemplazar esa dirección Ethernet con la del agente de base se hace por un truco llamado **ARP gratuito.** Éste es un mensaje especial, no solicitado al enrutador que hace reemplazar una entrada específica del caché, en este caso la del *host* móvil que está a punto de salir. Cuando el *host* móvil vuelve después, se usa el mismo truco para actualizar de nuevo el caché del enrutador.

Nada en el diseño le impide a un *host* móvil ser su propio agente foráneo, pero ese método sólo funciona si el *host* móvil (en su capacidad de agente foráneo) está conectado lógicamente a Internet

en su sitio actual. Incluso, el *host* móvil debe tener la capacidad de adquirir una dirección IP (temporal). Esa dirección IP debe pertenecer a la LAN a que está adjunto actualmente.

La solución de la IETF para los *hosts* móviles resuelve varios otros problemas no mencionados hasta ahora. Por ejemplo, ¿cómo se localizan agentes? La solución es que cada agente transmita periódicamente su dirección y el tipo de servicios que está dispuesto a proporcionar (por ejemplo, de base, foráneo, o ambos). Cuando un *host* móvil llega a alguna parte, simplemente puede escuchar estas difusiones, llamadas **anuncios.** Como alternativa, puede difundir un paquete que anuncie su llegada y esperar que el agente foráneo local responda a él.

Otro problema que se tuvo que resolver es qué hacer con los *host* móviles mal educados que se van sin decir adiós. La solución es hacer válido el registro sólo para un intervalo de tiempo fijo. Si no se actualiza periódicamente, queda fuera para que el *host* foráneo pueda limpiar sus tablas.

Otro problema más es la seguridad. Cuando un agente de base recibe un mensaje que le pide que por favor envíe todos los paquetes de Roberta a alguna dirección IP, lo mejor es no hacerlo a menos que se convenza que Roberta es el origen de esta solicitud, y no alguien que intenta personificarla. Para este propósito se usan los protocolos de autenticación criptográficos. En el capítulo 8 estudiaremos esos protocolos.

Un punto final abordado por el grupo de trabajo se relaciona con los niveles de movilidad. Imagine un avión con una Ethernet a bordo utilizada por la navegación y computadoras de la aviación. En esta Ethernet hay un enrutador normal que se comunica con la Internet conectada en tierra a través de un enlace de radio. Un buen día, algún hábil ejecutivo de marketing tiene la idea de instalar conectores de Ethernet en todos los descansabrazos para que los pasajeros con computadoras móviles también se puedan conectar.

Ahora tenemos dos niveles de movilidad: las propias computadoras del avión que son estacionarias con respecto a Ethernet y las de los pasajeros, que son móviles con respecto al avión. Además, el enrutador a bordo es móvil con respecto a los enrutadores en tierra. Al ser móvil con respecto a un sistema que de suyo es móvil, se puede manejar utilizando el entunelamiento recursivo.

5.6.8 IPv6

Si bien el CIDR y el NAT pueden durar unos pocos años más, todo mundo se da cuenta que los días del IP en su forma actual (Ipv4) están contados. Además de estos problemas técnicos, hay otro asunto acechando. Hasta hace poco, la Internet ha sido utilizada en gran medida por universidades, industrias de alta tecnología y el gobierno (especialmente el Departamento de Defensa de Estados Unidos). Con la explosión del interés por la Internet que comenzó a mediados de la década de 1990, es muy posible que en el próximo milenio será usada por un grupo mucho más grande de gente, especialmente gente con necesidades diferentes. Por una parte, millones de personas con computadoras portátiles inalámbricas podrían usarla para mantenerse en contacto con sus bases. Por otra, con la inminente convergencia de las industrias de la computación, las comunicaciones y el entretenimiento, tal vez no pase mucho tiempo ante de que todos los teléfonos y los televisores del

mundo estén en un nodo Internet, produciendo mil millones de máquinas utilizadas para recibir audio y vídeo bajo demanda. En estas circunstancias, se hizo evidente que el IP tenía que evolucionar y volverse más flexible.

Al ver en el horizonte estos problemas, la IETF comenzó a trabajar en 1990 en una versión nueva del IP, una que nunca se quedaría sin direcciones, resolvería varios otros problemas y sería más flexible y eficiente también. Sus metas principales eran:

1. Manejar miles de millones de *hosts*, aún con asignación de espacio de direcciones ineficiente.
2. Reducir el tamaño de las tablas de enrutamiento.
3. Simplificar el protocolo, para permitir a los enrutadores el procesamiento más rápido de los paquetes.
4. Proporcionar mayor seguridad (verificación de autenticidad y confidencialidad) que el IP actual.
5. Prestar mayor atención al tipo de servicio, especialmente con datos en tiempo real.
6. Ayudar a la multidifusión permitiendo la especificación de alcances.
7. Posibilitar que un *host* sea móvil sin cambiar su dirección.
8. Permitir que el protocolo evolucione.
9. Permitir que el protocolo viejo y el nuevo coexistan por años.

Para encontrar un protocolo que cumpliera con todos estos requisitos, la IETF hizo una convocatoria solicitando propuestas y estudios en el RFC 1550. Se recibieron 21 respuestas, no todas propuestas completas. En diciembre de 1992 había siete propuestas serias en la mesa; iban desde hacer cambios menores al IP hasta desecharlo y reemplazarlo por un protocolo completamente diferente.

Una propuesta fue ejecutar TCP sobre CLNP, que, con sus direcciones de 160 bits habría proporcionado suficiente espacio de direcciones para siempre y habría unificado dos protocolos de capa de red principales. Sin embargo, muchos pensaron que esto habría sido una aceptación de que algunas cosas en el mundo OSI en realidad estaban bien hechas, algo considerado políticamente incorrecto en los círculos de Internet. El CLNP se creó tomando como modelo al IP, por lo que los dos no son realmente tan diferentes. De hecho, el protocolo escogido es más diferente del IP que CLNP. Otra cosa en contra de CLNP fue su pobre manejo de tipos de servicio, algo requerido para difundir multimedia eficientemente.

Tres de las mejores propuestas se publicaron en *IEEE Network* (Deering, 1993; Francis, 1993, y Katz y Ford, 1993). Tras muchos análisis, revisiones e intrigas, se seleccionó una versión modificada de la combinación de las propuestas de Deering y Francis, llamada ahora **SIPP** (**Protocolo Simple de Internet Mejorado**), y se le dio la designación **Ipv6.**

El IPv6 cumple los objetivos bastante bien: mantiene las buenas características del IP, descarta y reduce las malas, y agrega nuevas donde se necesitan. En general, IPv6 no es compatible con

IPv4, pero es compatible con todos los demás protocolos Internet, incluidos TCP, UDP, ICMP, IGMP, OSPF, BGP y DNS, a veces con algunas pequeñas modificaciones (principalmente para manejar direcciones más grandes). Las características principales del IPv6 se analizan a continuación. Puede encontrarse mayor información en los RFCs 2460 a 2466.

Por principio, y lo más importante, el IPv6 tiene direcciones más grandes que el IPv4; son de 16 bytes de longitud, lo que resuelve el problema que se buscaba resolver: proporcionar una cantidad prácticamente ilimitada de direcciones Internet. Hablaremos más sobre las direcciones un poco más adelante.

La segunda mejora principal del IPv6 es la simplificación del encabezado, que contiene sólo 7 campos (contra 13 en el IPv4). Este cambio permite a los enrutadores procesar con mayor rapidez los paquetes y mejorar, por tanto, la velocidad real de transporte. También estudiaremos pronto el encabezado.

La tercera mejora importante fue el mejor apoyo de las opciones. Este cambio fue esencial con el nuevo encabezado, pues campos que antes eran obligatorios ahora son opcionales. Además, es diferente la manera de representar las opciones, haciendo más sencillo que los enrutadores hagan caso omiso de opciones no dirigidas a ellos. Esta característica mejora el tiempo de procesamiento de los paquetes.

Una cuarta área en la que el IPv6 representa un avance importante es la seguridad. La IETF tenía infinidad de historias sobre preadolescentes precoces que usaban sus computadoras personales para meterse en bancos e instalaciones militares por todas partes de Internet. Se tenía la fuerte sensación de que había que hacerse algo para mejorar la seguridad. La autenticación y la privacidad son características clave del IP nuevo. Estas características fueron incluídas posteriormente en el IPv4, así que las diferencias no son tan marcadas en el área de la seguridad.

Por último, se ha puesto mayor atención en la calidad del servicio. En el pasado se realizaron varios esfuerzos débiles, pero con el crecimiento actual de la multimedia en Internet, se requiere un mayor esfuerzo.

El encabezado principal del IPv6

El encabezado del IPv6 se muestra en la figura 5-68. El campo de *Versión* siempre es 6 para el IPv6 (y de 4 para el IPv4). Durante el periodo de transición del IPv4 al IPv6, que probablemente se llevará una década, los enrutadores podrán examinar este campo para saber el tipo de paquete que tienen. Como nota al margen, esta prueba ocupa algunas instrucciones en la ruta crítica, por lo que muchas implementaciones probablemente la evitarán usando algún campo del encabezado de enlace de datos para distinguir los paquetes IPv4 de los IPv6. De esta manera, los paquetes pueden pasarse directamente al manejador correcto de la capa de red. Sin embargo, hacer que la capa de enlace de datos esté consciente de los tipos de los paquetes de red viola por completo el principio de diseño de que ninguna capa debe estar enterada del significado de los bits entregados por la capa superior a ella. Los debates entre los bandos de “hacerlo bien” y “hacerlo rápido” sin duda serán largos y acalorados.

El campo *Clase de tráfico* se usa para distinguir entre los paquetes con requisitos diferentes de entrega en tiempo real. Un campo diseñado para este propósito ha estado en el IP desde el principio,

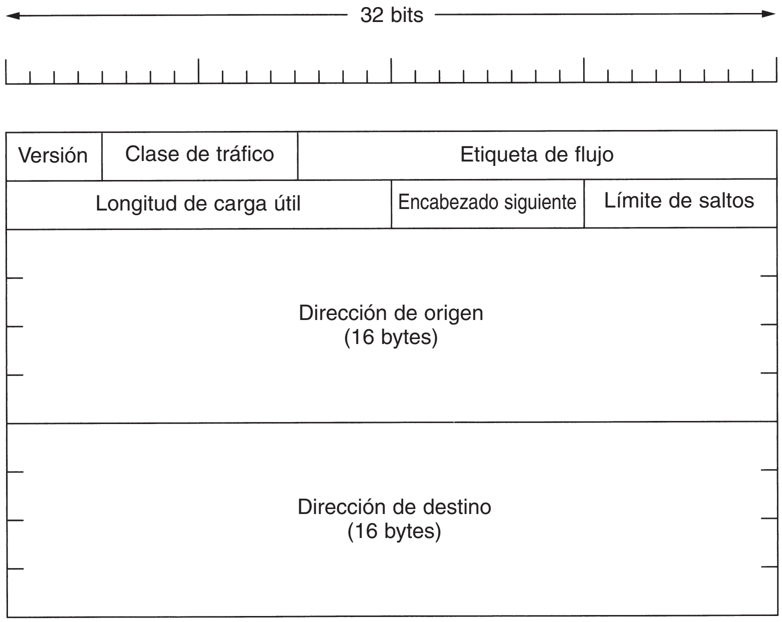


Figura 5-68. Encabezado fijo del IPv6 (obligatorio).

pero los enrutadores lo han implementado sólo esporádicamente. Ahora están en camino los experimentos para determinar qué tan bueno puede ser para usarse en la entrega de multimedia.

El campo de *Etiqueta de flujo* aún es experimental, pero se usará para permitir a un origen y a un destino establecer una pseudoconexión con propiedades y requisitos particulares. Por ejemplo, una cadena de paquetes de un proceso en cierto *host* de origen dirigido a cierto proceso en cierto *host* de destino puede tener requisitos de retardo muy estrictos y, por tanto, necesitar un ancho de banda reservado. El flujo puede establecerse por adelantado, dándole un identificador. Cuando aparece un paquete con una *Etiqueta de flujo* diferente de cero, todos los enrutadores pueden buscarla en sus tablas internas para ver el tipo de tratamiento especial que requiere. En efecto, los flujos son un intento de tener lo mejor de ambos mundos: la flexibilidad de una subred de datagramas y las garantías de una subred de circuitos virtuales.

Cada flujo está designado por la dirección de origen, la dirección de destino y el número de flujo, por lo que pueden estar activos muchos flujos al mismo tiempo entre un par dado de direcciones IP. También, de esta manera, aun si dos flujos provenientes de *hosts* diferentes pero con el mismo número de flujo pasan por el mismo enrutador, el enrutador será capaz de distinguirlos usando las direcciones de origen y destino. Se espera que se escojan los números de flujo al azar, en lugar de asignarlos secuencialmente comenzando por el 1, para simplificar el proceso de dispersión en los enrutadores.

El campo de *Longitud de carga útil* indica cuántos bytes siguen al encabezado de 40 bytes de la figura 5-68. El nombre de campo se cambió de *Longitud total* en el IPv4 porque el significado

cambió ligeramente: los 40 bytes del encabezado ya no se cuentan como parte de la longitud, como antes.

El campo *Encabezado siguiente* revela el secreto. La razón por la que pudo simplificarse el encabezado es que puede haber encabezados adicionales (opcionales) de extensión. Este campo indica cuál de los seis encabezados de extensión (actualmente), de haberlos, sigue a éste. Si este encabezado es el último encabezado de IP, el campo de *Encabezado siguiente* indica el manejador de protocolo de transporte (por ejemplo, TCP, UDP) al que se entregará el paquete.

El campo de *Límite de saltos* se usa para evitar que los paquetes vivan eternamente. En la práctica es igual al campo de *Tiempo de vida* del IPv4, es decir, un campo que se disminuye en cada salto. En teoría, en el IPv4 era un tiempo en segundos, pero ningún enrutador lo usaba de esa manera, por lo que se cambió el nombre para reflejar la manera en que se usa realmente.

Luego vienen los campos de *Dirección de origen* y *Dirección de destino.* La propuesta original de Deering, el SIP, usaba direcciones de 8, pero durante el periodo de revisión muchas personas sintieron que, con direcciones de 8 bytes, en algunas décadas el IPv6 se quedaría sin direcciones, y que con direcciones de 16 bytes nunca se acabarían. Otros argumentaban que 16 bytes era demasiado, mientras que unos más estaban a favor de usar direcciones de 20 bytes para hacerlas compatibles con el protocolo de datagramas de OSI. Otro grupo quería direcciones de tamaño variable. Tras mucho debate, se decidió que la mejor media sería una dirección de 16 bytes de longitud fija.

Se ha desarrollado una nueva notación para escribir direcciones de 16 bytes: se escriben como ocho grupos de cuatro dígitos hexadecimales, separados los grupos por dos puntos, como sigue:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Ya que muchas direcciones tendrán muchos ceros en ellas, se han autorizado tres optimizaciones. Primero, los ceros a la izquierda de un grupo pueden omitirse, por lo que 0123 puede escribirse como 123. Segundo, pueden reemplazarse uno o más grupos de 16 ceros por un par de signos de dos puntos. Por tanto, la dirección anterior se vuelve ahora

8000::0123:4567:89AB:CDEF

Por último, las direcciones IPv4 pueden escribirse como un par de signos de dos puntos y un número decimal anterior separado por puntos, como por ejemplo

::192.31.20.46

Tal vez no sea necesario ser tan explícitos al respecto, pero hay muchas direcciones de 16 bytes. Específicamente, hay 2^{128} de ellas, lo que aproximadamente es 3×10^{38}. Si la Tierra completa, incluidos los océanos, estuviera cubierta de computadoras, el IPv6 permitiría 7×10^{23} direcciones IP por metro cuadrado. Los estudiantes de química notarán que este número es mayor que el número de Avogadro. Aunque no fue la intención darle a cada molécula de la superficie terrestre su propia dirección IP, no estamos lejos de ello.

En la práctica, el espacio de direcciones no se usará eficientemente, igual que el espacio de números telefónicos (el código de área de Manhattan, 212, está prácticamente lleno, pero el

de Wyoming, 307, está casi vacío). En el RFC 3194, Durand y Huitema calcularon que, usando la asignación de números telefónicos como guía, hasta en la situación más pesimista habrá más de 1000 direcciones IP por metro cuadrado de la superficie terrestre (tierra y agua). En cualquier situación probable, habrá billones de ellas por metro cuadrado. En pocas palabras, parece poco probable que se acabarán en el futuro previsible.

Es instructivo comparar el encabezado de IPv4 (figura 5-53) con el de IPv6 (figura 5-68) para ver lo que se ha dejado fuera del IPv6. El campo *IHL* se fue porque el encabezado de IPv6 tiene una longitud fija. El campo de *Protocolo* se retiró porque el campo *Encabezado siguiente* indica lo que sigue al último encabezado de IP (por ejemplo, un segmento UDP o TCP).

Se retiraron todos los campos relacionados con la fragmentación, puesto que el IPv6 tiene un enfoque distinto hacia la fragmentación. Para empezar, todos los *hosts* que se ajustan al IPv6 deben determinar dinámicamente el tamaño de datagrama. Asimismo, el mínimo se incrementó de 576 a 1280 para permitir 1024 bytes de datos y varios encabezados. Esta regla hace que sea menos posible que ocurra la fragmentación. Además, cuando un *host* envía un paquete de IPv6 demasiado grande, en lugar de fragmentarlo, el enrutador que es incapaz de reenviarlo devuelve un mensaje de error. Este mensaje indica al *host* que divida todos los paquetes futuros a ese destino. Es mucho más eficiente hacer que el *host* envíe paquetes del tamaño correcto desde el principio que hacer que los enrutadores los fragmenten sobre la marcha.

Por último, el campo de *Suma de verificación* desaparece, porque su cálculo reduce en gran medida el desempeño. Con las redes confiables de hoy, además del hecho de que la capa de enlace de datos y las capas de transporte normalmente tienen sus propias sumas de verificación, el provecho de otra suma de verificación no valía el costo de desempeño que generaba. Al removerse estas características ha quedado un protocolo de capa de red compacto y sólido. Por tanto, la meta del IPv6 (un protocolo rápido y flexible con bastante espacio de direcciones) se ha cumplido con este diseño.

Encabezados de extensión

Con todo, algunos de los campos faltantes del IPv4 en ocasiones son necesarios, por lo que el IPv6 introdujo el concepto de **encabezado de extensión** (opcional). Estos encabezados pueden usarse para proporcionar información extra, pero codificada de una manera eficiente. Hay seis tipos de encabezados de extensión definidos actualmente, que se listan en la figura 5-69. Todos son opcionales, pero si hay más de uno, deben aparecer justo después del encabezado fijo, y de preferencia en el orden listado.

Algunos de los encabezados tienen un formato fijo; otros contienen un número variable de campos de longitud variable. En éstos, cada elemento está codificado como una tupla (tipo, longitud, valor). El *Tipo* es un campo de 1 byte que indica la opción de la que se trata. Los valores de *Tipo* se han escogido de modo que los dos primeros bits indican a los enrutadores que no saben cómo procesar la opción lo que tienen que hacer. Las posibilidades son: saltar la opción, descartar el paquete, descartar el paquete y enviar de regreso un paquete ICMP, y lo mismo que lo anterior, pero no enviar paquetes ICMP a direcciones de multidifusión (para evitar que un paquete de multidifusión malo genere millones de informes ICMP).

Encabezado de extensión	Descripción
Opciones salto por salto	Información diversa para los enrutadores
Opciones de destino	Información adicional para el destino
Enrutamiento	Ruta total o parcial a seguir
Fragmentación	Manejo de fragmentos de datagramas
Autenticación	Verificación de la identidad del emisor
Carga útil de seguridad encriptada	Información sobre el contenido encriptado

Figura 5-69. Encabezados de extensión del IPv6.

La *Longitud* también es un campo de 1 byte, e indica la longitud del valor (0 a 255 bytes). El *Valor* es cualquier información requerida, de hasta 255 bytes.

El encabezado de salto por salto se usa para información que deben examinar todos los enrutadores a lo largo de la ruta. Hasta ahora, se ha definido una opción: manejo de datagramas de más de 64K. El formato de este encabezado se muestra en la figura 5-70. Cuando se utiliza, el campo de *Longitud* de carga útil del siguiente encabezado se establece a cero.

Encabezado siguiente	0	194	4
Longitud de la carga útil grande			

Figura 5-70. Encabezado de extensión salto por salto para datagramas grandes (jumbogramas).

Como todos los encabezados de extensión, éste comienza con 1 byte que indica el tipo de encabezado que sigue. A este byte le sigue uno que indica la longitud del encabezado salto por salto en bytes, excluyendo los primeros 8 bytes, que son obligatorios. Todas las extensiones empiezan de esta manera.

Los 2 bytes siguientes indican que esta opción define el tamaño del datagrama (código 194) como número de 4 bytes. Los últimos 4 bytes indican el tamaño del datagrama. No se permiten los tamaños menores que 65,536, que darán como resultado que el primer enrutador descarte el paquete y devuelva un mensaje ICMP de error. Los datagramas que usan este encabezado de extensión se llaman **jumbogramas.** El uso de jumbogramas es importante para las aplicaciones de supercomputadoras que deben transferir con eficiencia gigabytes de datos a través de Internet.

El encabezado de opciones de destino está proyectado para campos que sólo necesitan ser interpretados en el *host* de destino. En la versión inicial de IPv6, las únicas opciones definidas son las opciones nulas con respecto a inflar este encabezado a un múltiplo de 8 bytes, por lo que inicialmente no se usará. Se incluyó para asegurarse que ese nuevo enrutamiento y el software del *host* pudieran manejarlo, en caso de que algún día alguien pensara en una opción de destino.

El encabezado de enrutamiento lista uno o más enrutadores que deben visitarse en el camino al destino. Es muy similar al enrutamiento libre del IPv4 en el sentido de que todas las direcciones listadas se deben visitar en orden, pero también se podrían visitar otros enrutadores no listados que se encuentren en medio de la ruta. El formato del encabezado de enrutamiento se muestra en la figura 5-71.

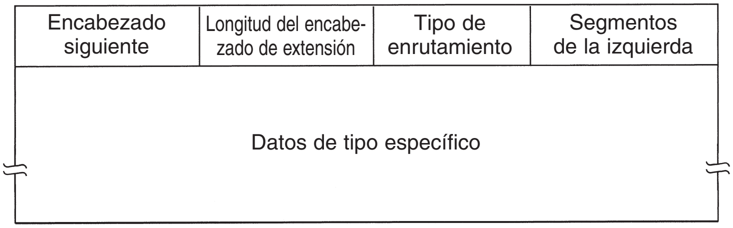


Figura 5-71. Encabezado de extensión para enrutamiento.

Los primeros 4 bytes del encabezado de extensión de enrutamiento contienen cuatro enteros de 1 byte. Anteriormente describimos los campos *Encabezado siguiente* y *Longitud del encabezado de extensión*. El campo *Tipo de enrutamiento* da el formato del resto del encabezado. Teclear 0 significa que una palabra reservada de 32 bits sigue a la primera palabra, seguida por algún número de direcciones de IPv6. Pueden inventarse otros tipos en el futuro según se necesite. Finalmente, el campo *Segmentos restantes* registra cuántas direcciones de la lista no se han visitado todavía. Se reduce cada vez que se visita una. Cuando llega a 0, el paquete está solo sin más guía sobre qué ruta seguir. En general, a estas alturas está tan cerca del destino que la mejor ruta es obvia.

El encabezado de fragmento maneja la fragmentación de una manera parecida a la del IPv4. El encabezado contiene el identificador del datagrama, el número de fragmento y un bit que indica si seguirán más fragmentos. En el IPv6, a diferencia del IPv4, sólo el *host* de origen puede fragmentar un paquete. Los enrutadores a lo largo del camino no pueden hacerlo. Aunque este cambio es un rompimiento filosófico con el pasado, simplifica el trabajo del enrutador y acelera el enrutamiento. Como se mencionó antes, si un enrutador confronta un paquete demasiado grande, lo descarta y devuelve un paquete ICMP al origen. Esta información permite que el *host* de origen fragmente el paquete en pedazos más pequeños usando este encabezado y lo intente de nuevo.

El encabezado de autenticación proporciona un mecanismo mediante el cual el receptor de un paquete puede estar seguro de quién lo envió. La carga útil de seguridad encriptada posibilita encriptar el contenido de un paquete de modo que sólo el receptor pretendido pueda leerlo. Estos encabezados usan técnicas criptográficas para lograr su cometido.

Controversias

Dados el proceso de diseño abierto y las fuertes opiniones de muchas de las personas participantes, no debería sorprender que muchas decisiones tomadas para el IPv6 fueran tema de fuertes controversias. Resumiremos a continuación algunas de ellas. Para los detalles, vea los RFCs.

Ya hemos mencionado el argumento sobre la longitud de las direcciones. El resultado fue una solución intermedia: las direcciones de 16 bytes de longitud fija.

Surgió otra pelea sobre la longitud del campo de *Límite de saltos*. Una parte sentía que la limitación de la cantidad máxima de saltos a 255 (implícita al usar un campo de 8 bits) fue un error grave. A fin de cuentas, hoy son comunes las rutas de 32 saltos, y en 10 años podrán ser comunes rutas mucho más grandes. Esta gente argumentaba que el uso de una dirección enorme era ir demasiado lejos, pero que el uso de una cuenta de saltos pequeña era tener una visión miope. Desde su punto de vista, el peor pecado que puede cometer un informático es proporcionar insuficientes bits en algún lugar.

La respuesta fue que se pueden hacer argumentos para aumentar todos los campos, lo que llevaría a un encabezado inflamado. También, la función del campo de *Límite de saltos* es evitar que los paquetes vaguen durante demasiado tiempo, y 65,535 saltos son demasiados. Por último, a medida que crezca Internet, se construirán más y más enlaces de larga distancia, posibilitando la ida de un país a otro en media docena de saltos, cuando mucho. Si se requieren más de 125 saltos para llegar del origen y el destino a sus puertas de enlace internacionales, algo está mal con las redes dorsales nacionales. Los de 8 bits ganaron esta partida.

Otra papa caliente fue el tamaño máximo de paquete. La comunidad de las supercomputadoras quería paquetes de más de 64 KB. Cuando una supercomputadora comienza a transferir, el asunto realmente va en serio, y no quiere que se le interrumpa cada 64 KB. El argumento en contra de los paquetes grandes es que, si un paquete de 1 MB llega a una línea T1 de 1.5 Mbps, el paquete bloqueará la línea durante más de 5 segundos, produciendo un retardo muy notorio para los usuarios interactivos que comparten la línea. Se llegó a un punto medio: los paquetes normales se limitan a 64 KB, pero puede usarse el encabezado de extensión de salto por salto para permitir los jumbogramas.

Un tercer tema candente fue la desaparición de la suma de verificación del IPv4. Para algunas personas esto representa algo parecido a quitarle los frenos a un automóvil. Hacerlo ciertamente aligera al automóvil y, por tanto, puede ir más rápido pero, de ocurrir un evento inesperado, tendremos problemas.

El argumento en contra de las sumas de verificación fue que cualquier aplicación a la que de verdad le importa la integridad de sus datos de todos modos tiene que tener una suma de verificación en la capa de transporte, por lo que tener otra en el IP (además de la suma de verificación de la capa de enlace de datos) es un exceso. Además, la experiencia mostraba que el cálculo de una suma de verificación en el IP era un gasto importante en el IPv4. El bando en contra de la suma de verificación ganó ésta, y el IPv6 no tiene una suma de verificación.

Los *hosts* móviles también fueron tema de contienda. Si una computadora portátil vuela al otro lado del mundo, ¿puede continuar operando en el destino con la misma dirección IPv6, o tiene que usar un esquema con agentes foráneos y agentes de base? Los *hosts* móviles también generan asimetrías en el sistema de enrutamiento. Es posible el caso en que una computadora móvil pequeña pueda escuchar fácilmente la señal emitida por un enrutador estacionario grande, pero que el enrutador estacionario no pueda escuchar la débil señal emitida por el *host* móvil. En consecuencia, algunas personas querían incluir soporte explícito de *hosts* móviles en el IPv6. Este esfuerzo falló cuando no se pudo generar consenso para ninguna propuesta específica.

Probablemente la batalla principal fue sobre la seguridad. Todos estaban de acuerdo en que se necesitaba. La guerra fue sobre dónde y cuándo. Primero dónde. El argumento a favor de ponerla en la capa de red es que entonces se vuelve un servicio estándar que todas las aplicaciones pueden usar sin ninguna planeación adelantada. El argumento en contra es que las aplicaciones realmente seguras por lo general no quieren nada menos que la encriptación de terminal a terminal, donde la aplicación de origen hace la encriptación y la aplicación de destino la deshace. Con cualquier otra cosa menos, el usuario está a merced de implementaciones de capa de red con fallas potenciales, sobre las que no tiene control. La respuesta a este argumento es que tales aplicaciones simplemente pueden abstenerse de usar las características de seguridad del IP y encargarse ellas mismas del asunto. La réplica a esto es que la gente que no confía en que la red lo haga bien no quiere pagar el precio de implementaciones de IP lentas y estorbosas que tengan esta capacidad, aun si está inhabilitada.

Otro aspecto sobre dónde poner la seguridad se relaciona con que muchos países (pero no todos) tienen leyes de exportación estrictas en lo referente a criptografía. Algunos, particularmente Francia e Irak, también restringen mucho su uso doméstico, de manera que la gente no pueda ocultar secretos de la policía. Como resultado, cualquier implementación de IP que use un sistema criptográfico lo bastante robusto como para tener algún valor no podría exportarse de los Estados Unidos (y de muchos otros países) a clientes mundiales. La mayoría de los proveedores de computadoras se oponen enérgicamente a tener que mantener dos juegos de *software*, uno para uso doméstico y otro para exportación.

Un punto donde no hubo controversia es que nadie espera que la Internet IPv4 se apague un domingo por la mañana y reinicie como Internet IPv6 la mañana del lunes. En cambio, se convertirían "islas" aisladas a IPv6, comunicándose inicialmente a través de túneles. A medida que crezcan las islas IPv6, se integrarán a islas más grandes. Tarde o temprano todas las islas se integrarán, y la Internet habrá sido convertida por completo. Dada la cuantiosa inversión en los enrutadores IPv4 actualmente instalados, el proceso de conversión probablemente tardará una década. Por esta razón, se ha puesto un enorme esfuerzo en asegurar que esta transición sea lo menos dolorosa posible. Para mayor información sobre IPv6, vea (Loshin, 1999).

5.7 RESUMEN

La capa de red proporciona servicios a la capa de transporte; puede basarse tanto en circuitos virtuales como en datagramas. En ambos casos, la tarea principal de esta capa es enrutar paquetes del origen al destino. En las subredes de circuitos virtuales se toma una decisión de enrutamiento al establecerse el circuito virtual; en las subredes de datagramas, se hace en cada paquete.

Se usan muchos algoritmos de enrutamiento en las redes de computadoras. Los algoritmos estáticos incluyen el enrutamiento por ruta más corta y la inundación. Los algoritmos dinámicos incluyen el enrutamiento por vector de distancia y el enrutamiento por estado del enlace. La mayoría de las redes usan algunos de éstos. Otros temas importantes relativos al enrutamiento son el enrutamiento jerárquico, el enrutamiento para *hosts* móviles, el enrutamiento por difusión, el enrutamiento por multidifusión y el enrutamiento en redes de igual a igual.

Las subredes pueden congestionarse, aumentando el retardo y reduciendo la velocidad real de transporte de los paquetes. Los diseñadores de redes intentan evitar la congestión mediante un diseño adecuado. Las técnicas incluyen política de retransmisión, almacenamiento en caché, control de flujo, entre otras. Si ocurre congestión, habrá que encargarse de ella. Pueden enviarse paquetes reguladores de regreso, desecharse parte de la carga, y aplicarse otros métodos.

El siguiente paso que va más allá de sólo tratar con la congestión es tratar realmente de alcanzar una calidad prometida de servicio. Los métodos que se pueden utilizar para esto incluyen el almacenamiento en el búfer en el cliente, el modelado de tráfico, la reservación de recursos y el control de acceso. Entre los métodos que se han diseñado para una buena calidad del servicio se encuentran los servicios integrados (incluyendo RSVP), los servicios diferenciados y MPLS.

Las redes difieren de varias maneras, por lo que cuando se conectan múltiples redes pueden ocurrir problemas. A veces los problemas pueden superarse enviando los paquetes mediante túneles a través de una red distinta, pero si las redes de origen y destino son diferentes, este método falla. Cuando las diferentes redes tienen diferentes tamaños máximos de paquete, puede requerirse una fragmentación.

La Internet posee una copiosa variedad de protocolos relacionados con la capa de red. Éstos incluyen protocolo de capa de transporte de datos, IP, pero también los protocolos de control ICMP, ARP y RARP, y los protocolos de enrutamiento OSPF y BGP. Internet se está quedando sin direcciones IP, por lo que se ha desarrollado una versión nueva de IP, el IPv6.

PROBLEMAS

1. Indique dos aplicaciones de ejemplo para las cuales es adecuado un servicio orientado a conexiones. Luego dé dos ejemplos en los que el servicio sin conexiones es lo mejor.

2. ¿Hay circunstancias en las cuales un servicio de circuito virtual entregará (o cuando menos debería entregar) paquetes en desorden? Explique.

3. Las subredes de datagramas enrutan cada paquete como unidad separada, independiente de las demás. Las subredes de circuitos virtuales no tienen que hacer esto, ya que cada paquete de datos sigue una ruta predeterminada. ¿Significa esto que las subredes de circuitos virtuales no necesitan la capacidad de enrutar paquetes aislados de un origen arbitrario a un destino arbitrario? Explique su respuesta.

4. Dé tres ejemplos de parámetros de protocolo que podrían negociarse al establecer una conexión.

5. Considere el siguiente problema de diseño que concierne a la implementación del servicio de circuitos virtuales. Si los circuitos virtuales son internos a la subred, cada paquete de datos debe tener un encabezado de 3 bytes, y cada enrutador debe destinar hasta 8 bytes de almacenamiento para la identificación de circuitos. Si se usan datagramas de manera interna, se requieren encabezados de 15 bytes, pero no se requiere espacio de tabla en los enrutadores. La capacidad de transmisión cuesta 1 centavo para cada 10^6 bytes, por salto. Puede comprarse memoria de enrutamiento por 1 centavo por byte y se de-

precia en dos años (tomando en cuenta que la semana laboral es de 40 horas). Estadísticamente, la sesión promedio dura 1000 seg, tiempo durante el cual se transmiten 200 paquetes. El paquete medio requiere cuatro saltos. ¿Qué implementación es más económica, y por cuánto?

6. Suponiendo que todos los enrutadores y *hosts* están trabajando de manera adecuada y que el software de ambos está libre de errores, ¿hay alguna posibilidad, por pequeña que sea, de que un paquete sea entregado al destino equivocado?

7. Considere la red de la figura 5-7, pero ignore los pesos de las líneas. Suponga que dicha red utiliza la inundación como algoritmo de enrutamiento. Si un paquete enviado mediante *A* a *D* tiene una cuenta máxima de salto de 3, liste todas las rutas que éste tomará. También mencione cuántos saltos merecedores de ancho de banda realiza.

8. Dé una heurística sencilla para encontrar dos rutas a través de una red de origen dado a un destino dado que pueda sobrevivir a la pérdida de cualquier línea de comunicación (suponiendo que existen dos de tales rutas). Los enrutadores se consideran lo bastante confiables, por lo que no es necesario preocuparse por la posibilidad de caída de los enrutadores.

9. Considere la subred de la figura 5-13(a). Se usa enrutamiento por vector de distancia y acaban de llegar los siguientes vectores al enrutador *C*: de *B:* (5, 0, 8, 12, 6, 2); de *D:* (16, 12, 6, 0, 9, 10), y de *E:* (7, 6, 3, 9, 0, 4). Los retardos medios a *B*, *D* y *E* son 6, 3 y 5, respectivamente. ¿Cuál es la nueva tabla de enrutamiento de *C*? Indique tanto la línea de salida a usar como el retardo esperado.

10. Si en una red de 50 enrutadores los retardos se registran como números de 8 bits y se intercambian vectores de retardo dos veces por segundo, ¿qué ancho de banda por línea dúplex total es consumido por el algoritmo de enrutamiento distribuido? Suponga que cada enrutador tiene tres líneas a los demás enrutadores.

11. En la figura 5-14 el OR booleano de los dos grupos de bits ACF es de 111 en cada fila. ¿Es éste un mero accidente, o es cierto para todas las subredes en todas las circunstancias?

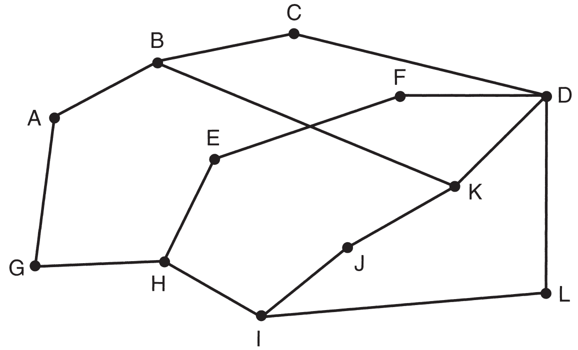
12. Para un enrutamiento jerárquico con 4800 enrutadores, ¿cuál región y tamaños de clúster deberían elegirse para minimizar el tamaño de la tabla de enrutamiento para una jerarquía de tres capas? Un buen lugar de inicio es la hipótesis de que una solución *k* clústeres de *k* regiones de *k* enrutadores está cerca de ser óptima, lo que significa que *k* es aproximadamente la raíz cúbica de 4800 (cerca de 16). Utilice la prueba y el error para verificar las combinaciones en las que los tres parámetros están en el límite de 16.

13. En el texto se indicó que, cuando un *host* móvil no está en casa, los paquetes enviados a su LAN base son interceptados por su agente de base en esa LAN. En una red IP de una LAN 802.3, ¿cómo logra esta intercepción el agente de base?

14. Viendo la subred de la figura 5-6, ¿cuántos paquetes se generan por una difusión de *B*, usando
(a) reenvío por ruta invertida?
(b) árbol sumidero?

15. Considere la red de la figura 5-16(a). Imagine que entre *F* y *G* se agrega una línea nueva pero el árbol sumidero de la figura 5-16(b) permanece sin cambios. ¿Qué cambios ocurren en la figura 5-16(c)?

16. Calcule un árbol de expansión multidifusión para el enrutador *C* de la siguiente subred para un grupo con miembros en los enrutadores *A*, *B*, *C*, *D*, *E*, *F*, *I* y *K*.



17. En la figura 5-20, ¿los nodos *H* o *I* difunden alguna vez en la búsqueda mostrada iniciada en *A*?

18. Suponga que el nodo *B* de la figura 5-20 ha reiniciado y no tiene información de enrutamiento en sus tablas. De repente necesita una ruta a *H*. Envía difusiones con *TTL* establecido a 1, 2, 3, etcétera. ¿Cuántas rondas da para encontrar la ruta?

19. En la versión más simple del algoritmo Chord para búsqueda de igual a igual, las búsquedas no utilizan la tabla *finger*. En su lugar, se ejecutan en forma lineal alrededor del círculo, en cualquier dirección. ¿Puede un nodo predecir de manera precisa en qué dirección debe buscar? Explique su respuesta.

20. Considere el círculo Chord de la figura 5-24. Suponga que el nodo 10 de repente se activa. ¿Esto afecta la tabla *finger* del nodo 1; de ser así, cómo?

21. Como posible mecanismo de control de congestión en una subred que usa circuitos virtuales internamente, un enrutador podría abstenerse de confirmar la recepción de un paquete hasta que (1) sabe que su última transmisión por el circuito virtual se recibió con éxito y que (2) tiene un búfer libre. Por sencillez, suponga que los enrutadores usan un protocolo de parada y espera y que cada circuito virtual tiene un búfer dedicado a él para cada destino del tráfico. Si se quieren *T* seg para transmitir un paquete (de datos o de confirmación de recepción) y hay *n* enrutadores en la ruta, ¿cuál es la velocidad con que se entregan paquetes al *host* de destino? Suponga que los errores de transmisión son poco frecuentes y que la conexión *host*-enrutador es infinitamente rápida.

22. Una subred de datagramas permite que los enrutadores puedan deshacerse de paquetes cuando lo necesiten. La probabilidad de que un enrutador descarte un paquete es de *p*. Considere el caso de un *host* de origen conectado al enrutador de origen, que está conectado al enrutador de destino, y por él al *host* de destino. Si cualquiera de los enrutadores descarta un paquete, el *host* de origen tarde o temprano termina la temporización e intenta de nuevo. Si tanto las líneas *host*-enrutador como enrutador-enrutador se cuentan como saltos, ¿cuál es la media de
(a) saltos que da un paquete por transmisión?
(b) transmisiones que hace un paquete?
(c) saltos requeridos por paquete recibido?

23. Describa dos diferencias principales entre el método de bit de advertencia y el método RED.

24. Cite una razón por la que el algoritmo de cubeta con goteo debe tener sólo un paquete por intervalo, independientemente del tamaño del paquete.

25. La variante de conteo de bytes del algoritmo de cubeta con goteo se usa en cierto sistema. La regla es que pueden enviarse por intervalo un paquete de 1024 bytes, dos paquetes de 512 bytes, etcétera. Indique una restricción seria de este sistema que no se menciona en el texto.

26. Una red ATM usa un esquema de cubeta con *tokens* para la conformación de tráfico. Se pone un *token* nuevo en la cubeta cada 5 μseg. Cada *token* cabe en una celda, que contiene 48 bytes de datos ¿Cuál es la tasa de datos máxima sustentable?

27. Una computadora de una red de 6 Mbps se regula mediante una cubeta con *tokens*. La cubeta con *tokens* se llena a razón de 1 Mbps. Inicialmente está llena a su capacidad máxima de 8 megabits. ¿Durante cuánto tiempo puede la computadora transmitir a 6 Mbps?

28. Imagine una especificación de flujo que tiene un tamaño máximo de paquete de 1000 bytes, una tasa de cubeta con *tokens* de 10 millones de bytes/seg, un tamaño de cubeta con *tokens* de 1 millón de bytes y una tasa máxima de transmisión de 50 millones de bytes/seg. ¿Cuánto tiempo puede durar una ráfaga a la velocidad máxima?

29. La red de la figura 5-37 utiliza RSVP con árboles de multidifusión para los *hosts* 1 y 2. Suponga que el *host* 3 solicita un canal de ancho de banda de 2 MB/seg para un flujo del *host* 1 y otro canal de ancho de banda de 1 MB/seg para un flujo del *host* 2. Al mismo tiempo, el *host* 4 solicita un canal de ancho de banda de 2 MB/seg para un flujo del *host* 1 y el *host* 5 solicita un canal de ancho de banda de 1 MB/seg para un flujo del *host* 2. ¿Cuánto ancho de banda se reservará para estas solicitudes en los enrutadores *A*, *B*, *C*, *E*, *H*, *J*, *K* y *L*?

30. La CPU de un enrutador puede procesar 2 millones de paquetes/seg. La carga que se le ofrece es 1.5 millones de paquetes/seg. Si una ruta del origen al destino contiene 10 enrutadores, ¿cuánto tiempo tardan las CPUs en encolar y dar servicio?

31. Considere el usuario de los servicios diferenciados con reenvío expedito. ¿Hay alguna garantía de que los paquetes expeditos experimenten un retardo más pequeño que los paquetes regulares? ¿Por qué sí o por qué no?

32. ¿Es necesaria la fragmentación en interredes de circuitos virtuales concatenados o sólo en los sistemas de datagramas?

33. El entunelamiento a través de una subred de circuitos virtuales concatenada es directo: el enrutador multiprotocolo en un extremo sólo establece un circuito virtual al otro extremo y pasa los paquetes a través de él. ¿El entunelamiento también puede utilizarse en las subredes de datagramas? ¿De ser así, cómo?

34. Suponga que el *host A* está conectado a un enrutador *R* 1, *R* 1 está conectado a otro enrutador, *R* 2, y *R* 2 está conectado al *host B*. Suponga que un mensaje TCP que contiene 900 bytes de datos y 20 bytes de encabezados TCP se pasa al código IP en el *host A* para entregarlo a *B*. Muestre los campos *Longitud total, Identificación*, *DF*, *MF* y *Desplazamiento del fragmento* del encabezado IP en cada paquete transmitido a través de los tres enlaces. Suponga que el enlace *A-R1* puede soportar un tamaño máximo de trama de 1024 bytes, así como un encabezado de trama de 14 bytes; el enlace *R1-R2* puede soportar un tamaño máximo de trama de 512 bytes, así como un encabezado de trama de 8 bytes, y el enlace *R2-B* puede soportar un tamaño máximo de trama de 512 bytes, incluyendo un encabezado de trama de 12 bytes.

35. Un enrutador está eliminando paquetes IP cuya longitud máxima (datos más encabezado) es de 1024 bytes. Suponiendo que los paquetes vivan por 10 seg, ¿cuál es la velocidad máxima a la que el enrutador puede operar sin el peligro de desbordar el espacio de números ID del datagrama IP?

36. Un datagrama IP que utiliza la opción *Enrutamiento de origen estricto* tiene que fragmentarse. ¿Cree que la opción se copia en cada fragmento, o con colocarlo en el primer fragmento es suficiente? Explique su respuesta.

37. Suponga que en lugar de usar 16 bits para la parte de red de una dirección clase B, se hubieran usado 20 bits. ¿Cuántas redes clase B habría?

38. Convierta la dirección de IP cuya representación hexadecimal es C22F1582 a notación decimal con puntos.

39. Una red en Internet tiene una máscara de subred de 255.255.240.0. ¿Cuál es la cantidad máxima de *hosts* que puede manejar?

40. Hay una gran cantidad de direcciones IP consecutivas, comenzando en 198.16.0.0. Suponga que cuatro organizaciones, *A*, *B*, *C* y *D*, solicitan 4000, 2000, 4000, y 8000 direcciones, respectivamente, y en ese orden. Dé la primera dirección asignada, la última dirección IP asignada y la máscara en la notación *w.x.y.z/s* para cada una de ellas.

41. Un enrutador acaba de recibir las siguientes nuevas direcciones IP: 57.6.96.0/21, 57.6.104.0/21, 57.6.112.0/21 y 57.6.120.0/21. Si todas éstas utilizan la misma línea de salida, ¿se pueden agregar? De ser así, ¿a qué? Si no, ¿por qué?

42. El conjunto de direcciones IP de 29.18.0.0 a 19.18.128.255 se ha agregado a 29.18.0.0/17. Sin embargo, hay un hueco de 1024 direcciones sin asignar de 29.18.60.0 a 29.18.63.255 que de repente se asignan a un *host* que utiliza una línea de salida diferente. Ahora es necesario dividir la dirección agregada en sus bloques constituyentes, agregar el nuevo bloque a la tabla y, después, ver si es posible alguna reagregación? Si no lo es, ¿qué se puede hacer en lugar de eso?

43. Un enrutador tiene las siguientes entradas (CIDR) en su tabla de enrutamiento:

Dirección/máscara	**Siguiente salto**
135.46.56.0/22	Interfaz 0
135.46.60.0/22	Interfaz 1
192.53.40.0/23	Enrutador 1
predeterminada	Enrutador 2

Para cada una de las siguientes direcciones IP, ¿qué hace el enrutador si llega un paquete con esa dirección?

(a) 135.46.63.10
(b) 135.46.57.14
(c) 135.46.52.2
(d) 192.53.40.7
(e) 192.53.56.7

44. Muchas compañías tienen la política de contar con dos (o más) enrutadores que conecten a la compañía a Internet para proporcionar alguna redundancia en caso de que una de ellas falle. ¿Esta política aún es posible con NAT? Explique su respuesta.

45. Usted explica el protocolo ARP a un amigo. Cuando usted termina su explicación, él dice: “Ya entiendo. ARP proporciona un servicio a la capa de red, por lo que es parte de la capa de enlace de datos”. ¿Qué le diría a su amigo?

46. ARP y RARP asignan direcciones de un espacio a otro. En este sentido, son similares. Sin embargo, sus implementaciones son esencialmente diferentes. ¿En qué aspecto fundamental son diferentes?

47. Describa una forma de reensamblar fragmentos IP en el destino.

48. La mayoría de los algoritmos de reensamble de datagramas IP tienen un temporizador para evitar que un fragmento perdido enlace búferes de reensamble por siempre. Suponga que un datagrama se divide en cuatro fragmentos. Los primeros tres fragmentos llegan y el cuarto se retrasa. En algún momento, el temporizador termina, por lo que se descartan los tres fragmentos de la memoria del receptor. Un poco más tarde, llega el último fragmento. ¿Qué se debería hacer con él?

49. Tanto en IP como en ATM, la suma de verificación cubre sólo el encabezado y no los datos. ¿Por qué supone que se eligió este diseño?

50. Una persona que vive en Boston viaja a Minneápolis, y lleva su computadora portátil. Para su sorpresa, la LAN de su destino en Minneápolis es una LAN IP inalámbrica, por lo que no tiene que conectarse. ¿Para que el correo electrónico y otro tipo de tráfico llegue de manera correcta aún es necesario todo el proceso con los agentes de base y foráneos?

51. IPv6 utiliza direcciones de 16 bytes. Si un bloque de 1 millón de direcciones se asigna cada picosegundo, ¿cuánto tardará la dirección?

52. El campo *Protocolo* utilizado en el encabezado IPv4 no está presente en el encabezado IPv6 fijo. ¿Por qué?

53. Cuando se introduce el protocolo IPv6, ¿tiene que cambiarse el protocolo ARP? De ser así, ¿los cambios son conceptuales o técnicos?

54. Escriba un programa para simular enrutamiento que utilice inundación. Cada paquete debe contener un contador que se decrementa en cada salto. Cuando el contador llega a cero, el paquete se descarta. El tiempo es discreto y cada línea maneja un paquete por intervalo de tiempo. Cree tres versiones del programa: todas las líneas están inundadas, todas las líneas, excepto la de entrada, están inundadas, y sólo las k mejores líneas (elegidas de manera estática) están inundadas. Compare la inundación con el enrutamiento determinista ($k = 1$) con base en el retardo y el ancho de banda utilizado.

55. Escriba un programa que simule una red de computadoras usando tiempo discreto. El primer paquete de cada cola de enrutador da un salto por intervalo de tiempo. Cada enrutador sólo tiene un número finito de búferes. Si un paquete llega y no hay espacio para él, se descarta y no se retransmite. En su lugar, hay un protocolo de extremo a extremo, lleno de terminaciones de temporización y paquetes de confirmación de recepción, que en algún momento regenera dicho paquete del enrutador de origen. Grafique la velocidad real de transporte de la red como una función del intervalo de terminación de temporizador de extremo a extremo, con parámetros de tasa de error.

56. Escriba una función para realizar el reenvío en un enrutador IP. El procedimiento tiene un parámetro, una dirección IP. También tiene acceso a una tabla global que consiste de un arreglo de tres variables. Cada arreglo contiene tres enteros: una dirección IP, una máscara de subred y la línea de salida a utilizar. La función usa CIDR para buscar la dirección IP en la tabla y regresa la línea a utilizar como su valor.

57. Utilice los programas *traceroute* (UNIX) o *tracert* (Windows) para trazar la ruta de su computadora a varias universidades de otros continentes. Haga una lista de los enlaces transoceánicos que ha descubierto. Algunos sitios para probar son:

www.berkeley.edu (California)
www.mit.edu (Massachusetts)
www.vu.nl (Amsterdam)
www.ucl.ac.uk (Londres)
www.usyd.edu.au (Sydney)
www.u-tokyo.ac.jp (Tokyo)
www.uct.ac.za (Cape Town)

6

LA CAPA DE TRANSPORTE

La capa de transporte no es una capa más. Es el corazón de toda la jerarquía de protocolos. La tarea de esta capa es proporcionar un transporte de datos confiable y económico de la máquina de origen a la máquina de destino, independientemente de la red o redes físicas en uso. Sin la capa de transporte, el concepto total de los protocolos en capas tendría poco sentido. En este capítulo estudiaremos en detalle la capa de transporte, incluidos sus servicios, diseño, protocolos y desempeño.

6.1 EL SERVICIO DE TRANSPORTE

En las siguientes secciones daremos una introducción al servicio de transporte. Veremos el tipo de servicio proporcionado a la capa de aplicación. Veremos el tipo de servicio que se proporciona a la capa de aplicación. Para que el tema del servicio de transporte quede claro, analizaremos dos conjuntos de primitivas de la capa de transporte. Primero analizaremos uno muy sencillo (e hipotético) para mostrar las ideas básicas. Después veremos la interfaz que se utiliza comúnmente en Internet.

6.1.1 Servicios proporcionados a las capas superiores

La meta fundamental de la capa de transporte es proporcionar un servicio eficiente, confiable y económico a sus usuarios, que normalmente son procesos de la capa de aplicación. Para lograr este objetivo, la capa de transporte utiliza los servicios proporcionados por la capa de red. El hardware

o software de la capa de transporte que se encarga del trabajo se llama **entidad de transporte,** la cual puede estar en el *kernel* (núcleo) del sistema operativo, en un proceso de usuario independiente, en un paquete de biblioteca que forma parte de las aplicaciones de red o en la tarjeta de red. En la figura 6-1 se ilustra la relación (lógica) entre las capas de red, transporte y aplicación.

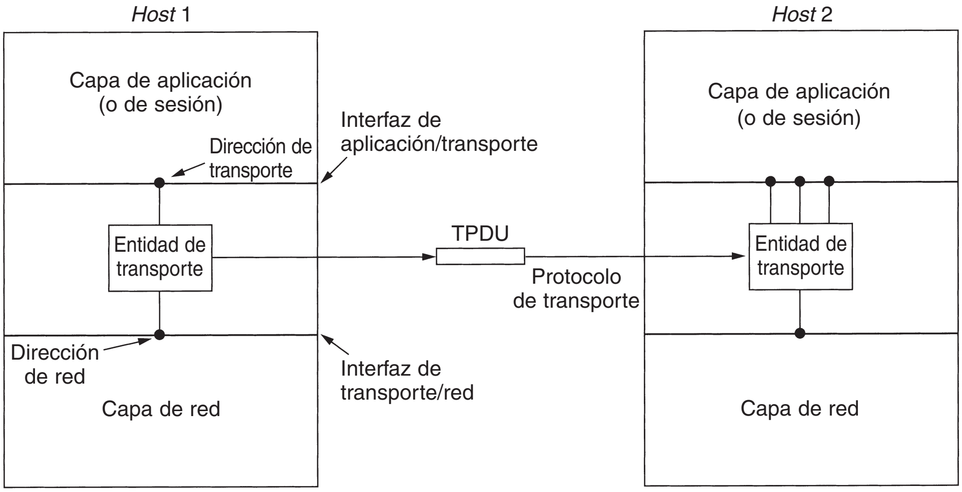


Figura 6-1. Las capas de red, transporte y aplicación.

Así como hay dos tipos de servicio de red, orientado y no orientado a la conexión, hay dos tipos de servicio de transporte. El servicio de transporte orientado a la conexión es parecido en muchos sentidos al servicio de red orientado a la conexión. En ambos casos, las conexiones tienen tres fases: establecimiento, transferencia de datos y liberación (o terminación). El direccionamiento y el control de flujo también son semejantes en ambas capas. Además, el servicio de transporte no orientado a la conexión es muy parecido al servicio de red no orientado a la conexión.

La pregunta obvia es: ¿si el servicio de la capa de transporte es tan parecido al de la capa de red, por qué hay dos capas diferentes? ¿Por qué no es suficiente una sola capa? La respuesta es sutil, pero crucial, y nos remite a la figura 1-9. El código de transporte se ejecuta por completo en las máquinas de los usuarios, pero la capa de red, por lo general, se ejecuta en los enrutadores, los cuales son operados por la empresa portadora (por lo menos en el caso de una red de área amplia). ¿Qué sucede si la capa de red ofrece un servicio poco confiable? ¿Qué tal si esa capa pierde paquetes con frecuencia? ¿Qué ocurre si los enrutadores se caen de cuando en cuando?

Problemas, eso es lo que ocurre. Los usuarios no tienen control sobre la capa de red, por lo que no pueden resolver los problemas de un mal servicio usando mejores enrutadores o incrementando el manejo de errores en la capa de enlace de datos. La única posibilidad es poner encima de la capa de red otra capa que mejore la calidad del servicio. Si, en una subred orientada a la conexión, a la mitad de una transmisión larga se informa a una entidad de transporte que su conexión de red ha sido terminada de manera abrupta, sin indicación de lo sucedido a los datos actualmente en tránsito, la entidad puede establecer una nueva conexión de red con la entidad de transporte

remota. Usando esta nueva conexión de red, la entidad puede enviar una solicitud a su igual preguntando cuáles datos llegaron y cuáles no, y reiniciar a partir de donde se originó la interrupción.

Esencialmente, la existencia de la capa de transporte hace posible que el servicio de transporte sea más confiable que el servicio de red subyacente. La capa de transporte puede detectar y compensar paquetes perdidos y datos alterados. Más aún, las primitivas del servicio de transporte se pueden implementar como llamadas a procedimientos de biblioteca con el propósito de que sean independientes de las primitivas del servicio de red, las cuales podrían variar considerablemente entre las redes (por ejemplo, el servicio LAN no orientado a la conexión puede ser bastante diferente del servicio WAN orientado a la conexión). Al ocultar el servicio de red detrás de un conjunto de primitivas de servicio de transporte, el cambio del servicio de red simplemente requiere reemplazar un conjunto de procedimientos de biblioteca por otro que haga lo mismo con un servicio subyacente distinto.

Gracias a la capa de transporte, es posible escribir programas de aplicación usando un conjunto estándar de primitivas, y que estos programas funcionen en una amplia variedad de redes sin necesidad de preocuparse por lidiar con diferentes interfaces de subred y transmisiones no confiables. Si ninguna red real tuviera fallas, y si todas tuvieran las mismas primitivas de servicio y se pudiera garantizar que nunca jamás cambiaran, tal vez la capa de transporte sería innecesaria. Sin embargo, en el mundo real esta capa cumple la función clave de aislar a las capas superiores de la tecnología, el diseño y las imperfecciones de la subred.

Por esta razón, mucha gente establece una distinción entre las capas 1 a 4, por una parte, y la(s) capa(s) por encima de la 4, por la otra. Las cuatro capas inferiores pueden verse como el **proveedor del servicio de transporte,** y la(s) capa(s) superiores son el **usuario del servicio de transporte.** Esta distinción entre proveedor y usuario tiene un impacto considerable en el diseño de las capas y pone a la capa de transporte en una posición clave, ya que constituye el límite principal entre el proveedor y el usuario del servicio confiable de transmisión de datos.

6.1.2 Primitivas del servicio de transporte

Para permitir que los usuarios accedan al servicio de transporte, la capa de transporte debe proporcionar algunas operaciones a los programas de aplicación, es decir, una interfaz del servicio de transporte. Cada servicio de transporte tiene su propia interfaz. Con el propósito de ver los aspectos básicos, en esta sección examinaremos primero un servicio de transporte sencillo (hipotético) y su interfaz. En la siguiente sección veremos un ejemplo real.

El servicio de transporte es parecido al servicio de red, pero hay algunas diferencias importantes. La principal es que el propósito del servicio de red es modelar el servicio ofrecido por las redes reales, con todos sus problemas. Las redes reales pueden perder paquetes, por lo que el servicio de red generalmente no es confiable.

En cambio, el servicio de transporte (orientado a la conexión) sí es confiable. Claro que las redes reales no están libres de errores, pero ése es precisamente el propósito de la capa de transporte: ofrecer un servicio confiable en una red no confiable.

Como ejemplo, considere dos procesos conectados mediante canalizaciones en UNIX. Ambos consideran que la conexión entre ellos es perfecta. No quieren saber de confirmaciones de recepción, paquetes perdidos, congestión ni nada por el estilo. Lo que quieren es una conexión 100 por ciento confiable. El proceso *A* pone datos en un extremo de la canalización y el proceso *B* los saca por el otro extremo. Ésta es la esencia del servicio de transporte orientado a la conexión: ocultar las imperfecciones del servicio de red para que los procesos usuarios puedan dar por hecho simplemente la existencia de un flujo de bits libre de errores.

Como nota al margen, la capa de transporte también puede proporcionar un servicio no confiable (de datagramas), pero hay muy poco que decir al respecto, por lo que en este capítulo nos concentraremos principalmente en el servicio de transporte orientado a la conexión. Sin embargo, hay algunas aplicaciones que se benefician del transporte no orientado a la conexión, como la computación cliente-servidor y la multimedia de flujo continuo, por lo que veremos algo sobre ellas más adelante.

Una segunda diferencia entre los servicios de red y de transporte es a quién están dirigidos. El servicio de red lo usan únicamente las entidades de transporte. Pocos usuarios escriben sus propias entidades de transporte y, por lo tanto, pocos usuarios o programas llegan a ver los aspectos internos del servicio de red. En contraste, muchos programas (y, por lo tanto, programadores) ven las primitivas de transporte. En consecuencia, el servicio de transporte debe ser adecuado y fácil de usar.

Para tener una idea de lo que podría ser el servicio de transporte, considere las cinco primitivas listadas en la figura 6-2. Esta interfaz de transporte ciertamente es sencilla, pero muestra la esencia de lo que debe hacer una interfaz de transporte orientada a la conexión: permite que los programas de aplicación establezcan, usen y liberen conexiones, lo cual es suficiente para muchas aplicaciones.

Primitiva	Paquete enviado	Significado
LISTEN	(ninguno)	Se bloquea hasta que algún proceso intenta la conexión
CONNECT	CONNECTION REQ.	Intenta activamente establecer una conexión
SEND	DATA	Envía información
RECEIVE	(ninguno)	Se bloquea hasta que llega un paquete DATA
DISCONNECT	DISCONNECTION REQ.	Este lado quiere liberar la conexión

Figura 6-2. Primitivas de un servicio de transporte sencillo.

Para ver cómo podrían usarse estas primitivas, considere una aplicación con un servidor y cierta cantidad de clientes remotos. Para comenzar, el servicio ejecuta una primitiva LISTEN, normalmente llamando a un procedimiento de biblioteca que hace una llamada de sistema para bloquear al servidor hasta la aparición de un cliente. Cuando un cliente desea comunicarse con el servidor, ejecuta una primitiva CONNECT. La entidad de transporte ejecuta esta primitiva bloqueando al

invocador y enviando un paquete al servidor. En la carga útil de este paquete se encuentra un mensaje de capa de transporte encapsulado, dirigido a la entidad de transporte del servidor.

Aquí es pertinente una nota rápida sobre la terminología. A falta de un mejor término, usaremos las siglas poco elegantes de **TPDU** (**Unidad de Datos del Protocolo de Transporte**) para referirnos a los mensajes enviados de una entidad de transporte a otra. Por lo tanto, las TPDUs (intercambiadas por la capa de transporte) están contenidas en paquetes (intercambiados por la capa de red). A su vez, los paquetes están contenidos en tramas (intercambiados por la capa de enlace de datos). Cuando llega una trama, la capa de enlace de datos procesa el encabezado de la trama y pasa el contenido del campo de carga útil de la trama a la entidad de red. Esta última procesa el encabezado del paquete y pasa el contenido de la carga útil del paquete a la entidad de transporte. Este anidamiento se ilustra en la figura 6-3.

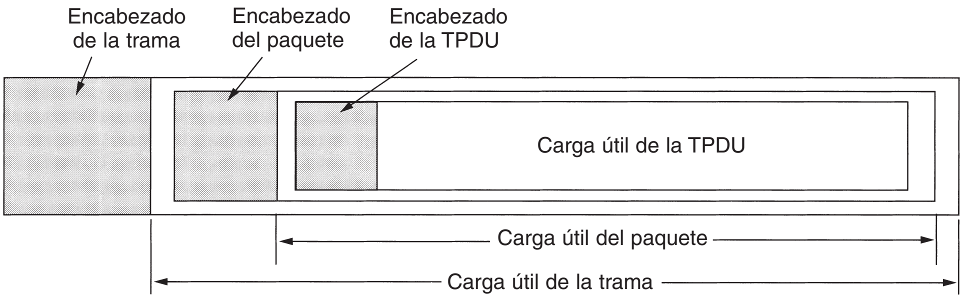


Figura 6-3. Anidamiento de las TPDUs, los paquetes y las tramas.

Regresando a nuestro ejemplo de cliente-servidor, la llamada CONNECT del cliente ocasiona el envío de una TPDU CONNECTION REQUEST (solicitud de conexión) al servidor. Al llegar ésta, la entidad de transporte verifica que el servidor esté bloqueado en LISTEN (es decir, esté interesado en manejar solicitudes). A continuación desbloquea el servidor y envía una TPDU CONNECTION ACCEPTED (conexión aceptada) de regreso al cliente. Al llegar esta TPDU, el cliente se desbloquea y se establece la conexión.

Ahora pueden intercambiarse datos usando las primitivas SEND y RECEIVE. En la forma más simple, cualquiera de las dos partes puede emitir una RECEIVE (bloqueadora) para esperar que la otra parte emita una SEND. Al llegar la TPDU, el receptor se desbloquea y puede procesar la TPDU y enviar una respuesta. Mientras ambos lados puedan llevar el control de quién tiene el turno para transmitir, este esquema funciona bien.

Observe que en la capa de transporte, incluso un intercambio de datos unidireccional es más complicado que en la capa de red. También se confirmará (tarde o temprano) la recepción de cada paquete de datos enviado. Asimismo, la recepción de los paquetes que llevan TPDUs de control se confirmará de manera implícita o explícita. Estas confirmaciones son manejadas por las entidades de transporte usando el protocolo de capa de red, y son transparentes para los usuarios de transporte. De la misma forma, las entidades de transporte necesitarán preocuparse por los temporizadores

y las retransmisiones. Los usuarios de transporte no se enteran de ningún aspecto de esta mecánica. Para ellos, una conexión es un conducto de bits confiable: un usuario mete bits en él y mágicamente aparecen en el otro lado. Esta capacidad de ocultar la complejidad es la razón por la cual los protocolos en capas son herramientas tan poderosas.

Cuando ya no se necesita una conexión, debe liberarse para desocupar espacio en las tablas de las dos entidades de transporte. La desconexión tiene dos variantes: asimétrica y simétrica. En la variante asimétrica, cualquiera de los dos usuarios de transporte puede emitir una primitiva DISCONNECT, que resulta en el envío de una TPDU DISCONNECT a la entidad de transporte remota. A su llegada, se libera la conexión.

En la variante simétrica, cada parte se cierra por separado, independientemente de la otra. Cuando una de las partes emite una DISCONNECT, quiere decir que ya no tiene más datos por enviar, pero aún está dispuesta a recibir datos de la otra parte. En este modelo, una conexión se libera cuando ambas partes han emitido una primitiva DISCONNECT.

En la figura 6-4 se presenta un diagrama de estado del establecimiento y liberación de una conexión con estas primitivas sencillas. Cada transición es activada por algún evento, ya sea una primitiva ejecutada por el usuario de transporte local o la llegada de un paquete. Por sencillez, aquí suponemos que la confirmación de recepción de cada TPDU se realiza por separado. También suponemos que se usa un modelo de desconexión simétrica, y que el cliente la realiza primero. Cabe señalar que este modelo es muy poco refinado. Más tarde veremos un modelo más realista.

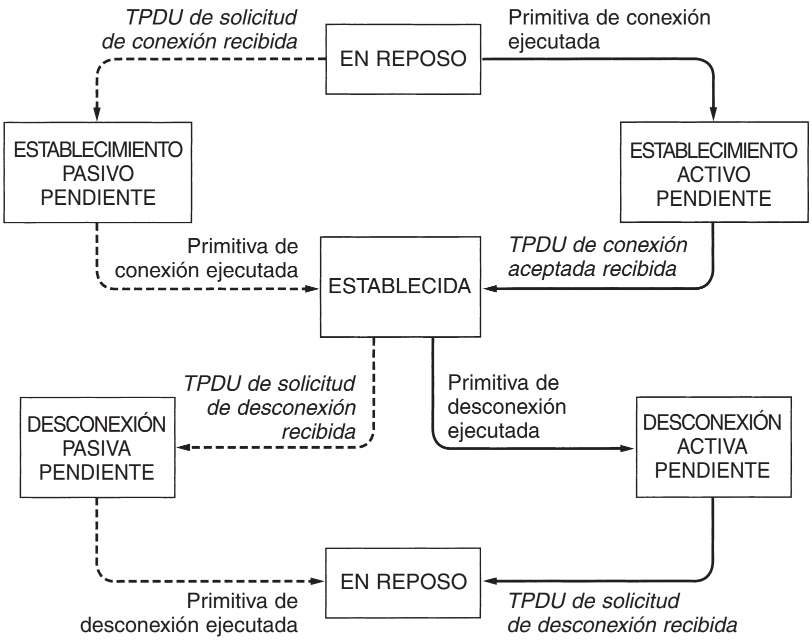


Figura 6-4. Diagrama de estado de un esquema sencillo de manejo de conexiones. Las transiciones escritas en cursivas son causadas por llegadas de paquetes. Las líneas continuas muestran la secuencia de estados del cliente. Las líneas punteadas muestran la secuencia de estados del servidor.

6.1.3 *Sockets* de Berkeley

Inspeccionemos brevemente otro grupo de primitivas de transporte, las primitivas de *socket* usadas en el UNIX de Berkeley para el TCP. Éstas se listan en la figura 6-5. En términos generales, las primitivas siguen el modelo de nuestro primer ejemplo, pero ofrecen más características y flexibilidad. No veremos aquí las TPDUs correspondientes. Ese análisis tendrá que esperar hasta que estudiemos el TCP posteriormente en este capítulo.

Primitiva	Significado
SOCKET	Crea un nuevo punto terminal de comunicación
BIND	Adjunta una dirección local a un *socket*
LISTEN	Anuncia la disposición a aceptar conexiones; indica el tamaño de cola
ACCEPT	Bloquea al invocador hasta la llegada de un intento de conexión
CONNECT	Intenta establecer activamente una conexión
SEND	Envía datos a través de la conexión
RECEIVE	Recibe datos de la conexión
CLOSE	Libera la conexión

Figura 6-5. Primitivas de *socket* para TCP.

Las primeras cuatro primitivas de la lista son ejecutadas en ese orden por los servidores. La primitiva SOCKET crea un nuevo punto de comunicación y le asigna espacio en las tablas de la entidad de transporte. Los parámetros de la llamada especifican el formato de direccionamiento que se utilizará, el tipo de servicio deseado (por ejemplo, flujo confiable de bytes) y el protocolo. Una llamada SOCKET con éxito devuelve un descriptor de archivo ordinario que se utiliza con las siguientes llamadas, de la misma manera que lo hace una llamada OPEN.

Los *sockets* recién creados no tienen direcciones de red. Éstas se asignan mediante la primitiva BIND. Una vez que un servidor ha destinado una dirección a un *socket*, los clientes remotos pueden conectarse a él. La razón para que la llamada SOCKET no cree directamente una dirección es que algunos procesos se encargan de sus direcciones (por ejemplo, han estado usando su misma dirección durante años y todos la conocen), mientras que otros no lo hacen.

A continuación viene la llamada LISTEN, que asigna espacio para poner en cola las llamadas entrantes por si varios clientes intentan conectarse al mismo tiempo. A diferencia de la llamada LISTEN de nuestro primer ejemplo, en el modelo de *sockets* LISTEN no es una llamada bloqueadora.

Para bloquearse en espera de una conexión entrante, el servidor ejecuta una primitiva ACCEPT. Cuando llega una TPDU solicitando una conexión, la entidad de transporte crea un *socket* nuevo con las mismas propiedades que el original y devuelve un descriptor de archivo para él. A continuación, el servidor puede ramificar un proceso o subproceso para manejar la conexión en el *socket* nuevo y regresar a esperar la siguiente conexión en el *socket* original. ACCEPT regresa un descriptor de archivo normal, que puede utilizarse para leer y escribir de la forma estándar, al igual que con los archivos.

Ahora veamos el cliente. Aquí también debe crearse un *socket* primero usando la primitiva SOCKET, pero no se requiere BIND, puesto que la dirección usada no le importa al servidor. La primitiva CONNECT bloquea al invocador y comienza activamente el proceso de conexión. Al completarse éste (es decir, cuando se recibe la TPDU adecuada del servidor) el proceso cliente se desbloquea y se establece la conexión. Ambos lados pueden usar ahora SEND y RECEIVE para transmitir y recibir datos a través de la conexión dúplex total. Las llamadas de sistema READ y WRITE de UNIX también se pueden utilizar si no son necesarias las opciones especiales de SEND y RECEIVE.

La liberación de las conexiones a los *sockets* es simétrica. La conexión se libera cuando ambos lados han ejecutado una primitiva CLOSE.

6.1.4 Un ejemplo de programación de *sockets*: un servidor de archivos de Internet

Veamos el código cliente-servidor de la figura 6-6 como ejemplo del uso de las llamadas de *sockets*. Ahí se muestra un servidor de archivos muy antiguo junto con un cliente de ejemplo que lo utiliza. El código tiene muchas limitaciones (que se analizan más adelante), pero en principio el código del servidor puede compilarse y ejecutarse en cualquier sistema UNIX conectado a Internet. A continuación, el código del cliente puede compilarse y ejecutarse en cualquier otra máquina UNIX conectada a Internet, en cualquier parte del mundo. El código del cliente puede ejecutarse con los parámetros apropiados para obtener cualquier archivo al que el servidor tenga acceso. El archivo se escribe a la salida estándar, la cual, por supuesto, puede redirigirse a un archivo o a un canal.

Veamos primero el código del servidor. Comienza con algunos encabezados estándar, los últimos tres de los cuales contienen las principales definiciones y estructuras de datos relacionadas con Internet. A continuación se encuentra una definición de *SERVER_PORT* como 12345. Este número se eligió de manera arbitraria. Cualquier número que se encuentre entre 1024 y 65535 funcionará siempre y cuando otro proceso no lo esté utilizando. Por supuesto, el cliente y el servidor tienen que utilizar el mismo puerto. Si este servidor llegara a convertirse en un éxito mundial (lo cual no es probable, debido a lo antiguo que es), se le asignaría un puerto permanente debajo de 1024 y aparecería en *www.iana.org*.

Las siguientes dos líneas del servidor definen dos constantes necesarias. La primera determina el tamaño de bloque utilizado para la transferencia de archivos. La segunda determina cuántas conexiones pendientes pueden almacenarse antes de empezar a descartar las excedentes que lleguen.

Después de las declaraciones de variables locales, comienza el código del servidor. Comienza inicializando una estructura de datos que contendrá la dirección IP del servidor. Esta estructura de datos pronto se anexará al *socket* del servidor. La llamada a *memset* establece en 0s toda la estructura de datos. Las siguientes tres asignaciones llenarán tres de sus campos. El último de éstos contiene el puerto del servidor. Las funciones *htonl* y *htons* están relacionadas con la conversión de valores a un formato estándar a fin de que el código se ejecute correctamente tanto en las máquinas *big-endian* (por ejemplo, la SPARC) como en las *little-endian* (por ejemplo, las Pentium). Su semántica exacta no es importante aquí.

A continuación el servidor crea un *socket* y verifica si hay errores (lo cual se indica mediante $s < 0$). En una versión de producción del código, el mensaje de error puede ser mucho más explicativo. La llamada a *setsockopt* es necesaria para permitir que el puerto sea reutilizado a fin de que el servidor se pueda ejecutar de manera indefinida, llenando los campos solicitud tras solicitud. Ahora la dirección IP se enlaza con el *socket* y se realiza una verificación para ver si la llamada a *bind* tuvo éxito. El último paso en la inicialización es la llamada a *listen* para anunciar que el servidor está dispuesto a aceptar llamadas entrantes e indicar al sistema que almacene la cantidad de ellas especificada en *QUEUE_SIZE* en caso de que lleguen más mientras el servidor aún esté procesando la actual. Si la cola está llena y llegan solicitudes adicionales, se descartan irremediablemente.

En este punto el servidor entra a su ciclo principal, al cual nunca abandona. La única forma de detenerlo es desde afuera. La llamada a *accept* bloquea el servidor hasta que algún cliente trata de establecer una conexión con él. Si la llamada a *accept* tiene éxito, se regresa un descriptor de archivo que puede utilizarse para leer y escribir, de la misma forma en la que los descriptores de archivo pueden utilizarse para leer y escribir desde las canalizaciones. Sin embargo, a diferencia de las canalizaciones, que son unidireccionales, los *sockets* son bidireccionales, por lo que *sa* (dirección de *socket*) puede utilizarse para leer de la conexión y también para escribir en ella.

Una vez que se establece la conexión, el servidor lee en ella el nombre del archivo. Si el nombre aún no está disponible, el servidor se bloquea y lo espera. Una vez que obtiene el nombre, el servidor abre el archivo y luego entra en un ciclo que lee de manera alterna bloques del archivo y los escribe en el *socket* hasta que el archivo se haya copiado por completo. A continuación el servidor cierra el archivo y la conexión y espera hasta que aparezca la siguiente conexión. Repite este ciclo de manera indefinida.

Ahora veamos el código del cliente. Para entender su funcionamiento, es necesario comprender cómo se invoca. Suponiendo que se llama *cliente*, una llamada típica es:

```
cliente flits.cs.vu.nl /usr/tom/nombredearchivo >f
```

Esta llamada sólo funciona si el servidor ya se está ejecutando en *flits.cs.vu.nl,* si existe el archivo */usr/tom/nombredearchivo* y si el servidor tiene acceso de lectura a él. Si la llamada es exitosa, el archivo se transfiere a través de Internet y se escribe en *f*, después de lo cual finaliza el programa cliente. Puesto que el servidor continúa después de una transferencia, el cliente puede iniciarse una y otra vez para obtener otros archivos.

El código del cliente inicia con algunas inclusiones y declaraciones. La ejecución comienza verificando si el código ha sido llamado con el número correcto de argumentos ($argc = 3$ significa el nombre del programa más dos argumentos). Observe que *argv* [1] contiene el nombre del servidor (por ejemplo, *flits.cs.vu.nl*) y que *gethostbyname* lo convierte en una dirección IP. Esta función utiliza DNS para buscar el nombre. En el capítulo 7 analizaremos DNS.

A continuación se crea e inicializa un *socket*. Después de esto, el cliente intenta establecer una conexión TCP con el servidor, mediante *connect*. Si el servidor está activo y ejecutándose en la máquina especificada y enlazado a *SERVER_PORT* y si está inactivo o tiene espacio es su cola *listen*, la conexión se establecerá (en algún momento). El cliente utiliza esta conexión para enviar el nombre del archivo escribiendo en el *socket*. El número de bytes enviados es un byte mayor que el propio nombre, puesto que se envía al servidor el byte 0 para indicarle en dónde termina dicho nombre.

```
/* Esta página contiene un programa cliente que puede solicitar un archivo
 * desde el programa servidor de la siguiente página. El servidor responde
 * enviando el archivo completo.
 */

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345                /* arbitrario, pero el cliente y el
                                         /* servidor deben coincidir */
#define BUF_SIZE 4096                    /* tamaño de bloque para transferencia */

int main(int argc, char **argv)
{
 int c, s, bytes;
 char buf[BUF_SIZE];                     /* búfer para el archivo entrante */
 struct hostent *h;                      /* información sobre el servidor */
 struct sockaddr_in channel;             /* contiene la dirección IP */

 if (argc != 3) fatal("Usage: client server-name file-name");
 h = gethostbyname(argv[1]);             /* busca la dirección IP del host */
 if (!h) fatal("gethostbyname failed");

 s = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
 if (s < 0) fatal("socket");
 memset(&channel, 0, sizeof(channel));
 channel.sin_family= AF_INET;
 memcpy(&channel.sin_addr.s_addr, h->h_addr, h->h_length);
 channel.sin_port= htons(SERVER_PORT);

 c = connect(s, (struct sockaddr *) &channel, sizeof(channel));
 if (c < 0) fatal("connect failed");

 /* Se ha establecido la conexión. Se envía el nombre del archivo incluyendo
 /* el byte 0 al final. */
  write(s, argv[2], strlen(argv[2])+1);

  /* Obtiene el archivo y lo escribe en la salida estándar. */
  while (1) {
      bytes = read(s, buf, BUF_SIZE);    /* lee del socket */
      if (bytes <= 0) exit(0);           /* verifica el final del archivo */
      write(1, buf, bytes);              /* escribe en la salida estándar */
 }
}

fatal(char *string)
{
 printf("%s\n", string);
 exit(1);
}
```

Figura 6-6. Código del cliente que utiliza *sockets*. El código del servidor se encuentra en la siguiente página.

```
#include <sys/types.h>                  /* Éste es el código del servidor */
#include <sys/fcntl.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345               /* arbitrario, pero el cliente y el
                                        /* servidor deben coincidir */
#define BUF_SIZE 4096                   /* tamaño de bloque para la
                                        /* transferencia */
#define QUEUE_SIZE 10

int main(int argc, char *argv[])
{
 int s, b, l, fd, sa, bytes, on = 1;
 char buf[BUF_SIZE];                    /* búfer para el archivo saliente */
 struct sockaddr_in channel;            /* contiene la dirección IP */

 /* Construye la estructura de la dirección para enlazar el socket. */
 memset(&channel, 0, sizeof(channel));    /* canal cero */
 channel.sin_family = AF_INET;
 channel.sin_addr.s_addr = htonl(INADDR_ANY);
 channel.sin_port = htons(SERVER_PORT);

 /* Apertura pasiva. Espera una conexión. */
 s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP); /* crea el socket */
 if (s < 0) fatal("socket failed");
 setsockopt(s, SOL_SOCKET, SO_REUSEADDR, (char *) &on, sizeof(on));

 b = bind(s, (struct sockaddr *) &channel, sizeof(channel));
 if (b < 0) fatal("bind failed");

 l = listen(s, QUEUE_SIZE);             /* especifica el tamaño de la cola */
 if (l < 0) fatal("listen failed");

 /* El socket ahora está configurado y enlazado. Espera una conexión y la
 /* procesa. */
 while (1) {
    sa = accept(s, 0, 0);               /* se bloquea para la solicitud de
                                        /* conexión */
    if (sa < 0) fatal("accept failed");

    read(sa, buf, BUF_SIZE);            /* lee el nombre del archivo desde el
                                        /* socket */

    /* Obtiene y regresa el archivo.*/
    fd = open(buf, O_RDONLY);           /* abre el archivo para regresarlo */
    if (fd < 0) fatal("open failed");

    while (1) {
         bytes = read(fd, buf, BUF_SIZE); /* lee del archivo */
         if (bytes <= 0) break;         /* verifica el final del archivo */
         write(sa, buf, bytes);         /* escribe bytes en el socket */
    }
    close(fd);                          /* cierra el archivo */
    close(sa);                          /* cierra la conexión */
 }
}
```

Ahora el cliente entra en un ciclo, lee el archivo bloque por bloque desde el *socket* y lo copia a la salida estándar. Cuando termina, simplemente abandona la conexión.

El procedimiento *fatal* imprime un mensaje de error y termina. El servidor necesita el mismo procedimiento, pero se omitió debido a la falta de espacio en la página. Puesto que el cliente y el servidor se compilan de manera separada y por lo general se ejecutan en computadoras diferentes, no pueden compartir el código de *fatal*.

Estos dos programas (así como otro material relacionado con este libro) se pueden obtener del sitio Web del libro

http://www.prenhall.com/tanenbaum

haciendo clic en el vínculo Companion Web Site que se encuentra junto a la fotografía de la portada. Dichos programas pueden bajarse y compilarse en cualquier sistema UNIX (por ejemplo, Solaris, BSD, Linux) mediante:

```
cc –o client client.c –lsocket -lnsl
cc –o server server.c –lsocket –lnsl
```

El servidor se inicia con sólo teclear

```
server
```

El cliente necesita dos argumentos, como se mencionó anteriormente. En el sitio Web también hay disponible una versión para Windows.

Sólo como aclaración, este servidor no tiene nada de refinado. Su verificación de errores es insuficiente y su reporte de errores es ordinario. Claramente, nunca ha escuchado sobre la seguridad, y utilizar sólo llamadas de sistema UNIX no es lo más recomendable respecto a independencia de la plataforma. También da por sentados algunos detalles que son técnicamente ilegales, como asumir que el nombre del archivo se ajusta en el búfer y que se transmite de manera inmediata y sin divisiones. Debido a que maneja todas las solicitudes en forma estrictamente secuencial (puesto que sólo tiene un solo subproceso), su desempeño es pobre. A pesar de estas fallas, es un servidor de archivos de Internet completo y funcional. En los ejercicios, se invita al lector a mejorarlo. Para mayor información sobre la programación con *sockets*, vea (Stevens, 1997).

6.2 ELEMENTOS DE LOS PROTOCOLOS DE TRANSPORTE

El servicio de transporte se implementa mediante un **protocolo de transporte** entre las dos entidades de transporte. En ciertos aspectos, los protocolos de transporte se parecen a los protocolos de enlace de datos que estudiamos detalladamente en el capítulo 3. Ambos se encargan del control de errores, la secuenciación y el control de flujo, entre otros aspectos.

Sin embargo, existen diferencias significativas entre los dos, las cuales se deben a diferencias importantes entre los entornos en que operan ambos protocolos, como se muestra en la figura 6-7. En la capa de enlace de datos, dos enrutadores se comunican directamente mediante un canal físico mientras que, en la capa de transporte, ese canal físico es reemplazado por la subred completa. Esta diferencia tiene muchas implicaciones importantes para los protocolos, como veremos en este capítulo.

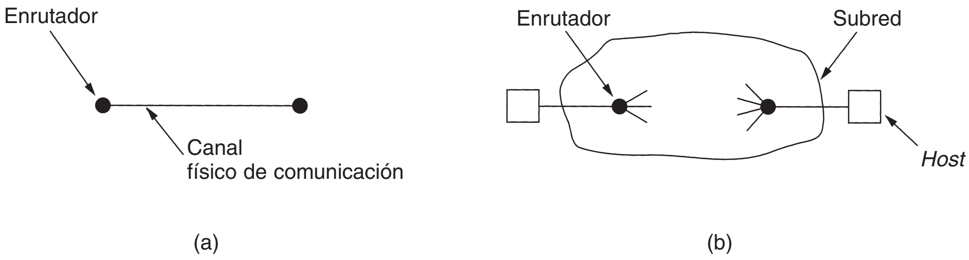


Figura 6-7. (a) Entorno de la capa de enlace de datos. (b) Entorno de la capa de transporte.

Por una parte, en la capa de enlace de datos no es necesario que un enrutador especifique el enrutador con el que quiere comunicarse; cada línea de salida especifica de manera única un enrutador en particular. En la capa de transporte se requiere el direccionamiento explícito de los destinos.

Por otro lado, el proceso de establecimiento de una conexión a través del cable de la figura 6-7(a) es sencillo: el otro extremo siempre está ahí (a menos que se caiga, en cuyo caso no estará ahí). Sea como sea, no hay demasiado que hacer. En la capa de transporte, el establecimiento inicial de la conexión es más complicado, como veremos.

Otra diferencia, muy irritante, entre la capa de enlace de datos y la capa de transporte es la existencia potencial de capacidad de almacenamiento en la subred. Al enviar un enrutador una trama, ésta puede llegar o perderse, pero no puede andar de un lado a otro durante un rato, esconderse en un rincón alejado del mundo y aparecer repentinamente en algún momento inoportuno 30 segundos después. Si la subred usa datagramas y enrutamiento adaptativo internamente, hay una probabilidad nada despreciable de que un paquete pueda almacenarse durante varios segundos y entregarse después. Las consecuencias de esta capacidad de almacenamiento de paquetes en la subred pueden ser desastrosas y requerir el uso de protocolos especiales.

Una última diferencia entre las capas de enlace de datos y de transporte es de cantidad, más que de tipo. Se requieren búferes y control de flujo en ambas capas, pero la presencia de una cantidad de conexiones grande y dinámicamente variable en la capa de transporte puede requerir un enfoque distinto del que se usa en la capa de enlace de datos. En el capítulo 3 vimos que algunos de los protocolos asignan una cantidad fija de búferes a cada línea de modo que, al llegar una trama, siempre hay un búfer disponible. En la capa de transporte, la gran cantidad de conexiones que deben manejarse hace menos atractiva la idea de dedicar muchos búferes a cada una. En las siguientes secciones examinaremos todos estos importantes temas, además de otros.

6.2.1 Direccionamiento

Cuando un proceso (por ejemplo, un usuario) de aplicación desea establecer una conexión con un proceso de aplicación remoto, debe especificar a cuál se conectará. (El transporte no orientado a la conexión tiene el mismo problema: ¿a quién debe enviarse cada mensaje?) El método que normalmente se emplea es definir direcciones de transporte en las que los procesos pueden estar a la

escucha de solicitudes de conexión. En Internet, estos puntos terminales se denominan **puertos.** En las redes ATM se llaman **AAL-SAPs.** Usaremos el término genérico **TSAP** (**Punto de Acceso al Servicio de Transporte**). Los puntos terminales análogos de la capa de red (es decir, direcciones de capa de red) se llaman **NSAP** (**Punto de Acceso al Servicio de Red**). Las direcciones IP son ejemplos de NSAPs.

En la figura 6-8 se ilustra la relación entre el NSAP, el TSAP y la conexión de transporte. Los procesos de aplicación, tanto clientes como servidores, se pueden enlazar por sí mismos a un TSAP para establecer una conexión a un TSAP remoto.Estas conexiones se realizan a través de NSAPs en cada *host,* como se muestra. Como en algunas redes cada computadora tiene un solo NSAP, los TSAPs sirven para distinguir los múltiples puntos terminales de transporte que comparten un NSAP.

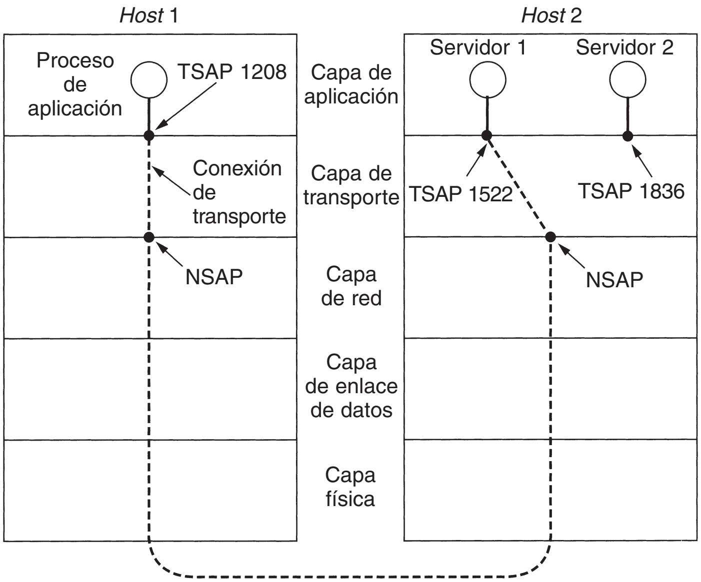


Figura 6-8. TSAPs, NSAPs y conexiones de transporte.

El siguiente es un posible escenario para una conexión de transporte.

1. Un proceso servidor de hora del día del *host* 2 se enlaza con el TSAP 1522 para esperar una llamada entrante. La manera en que un proceso se enlaza con un TSAP está fuera del modelo de red y depende por entero del sistema operativo local. Podría, por ejemplo, usarse una llamada como nuestra LISTEN.

2. Un proceso de aplicación del *host* 1 quiere averiguar la hora del día, por lo que emite una solicitud CONNECT especificando el TSAP 1208 como el origen y el TSAP 1522 como destino. Esta acción al final da como resultado una conexión de transporte que se establece entre el proceso de aplicación del *host* 1 y el servidor 1 del *host* 2.

3. A continuación el proceso de aplicación envía una solicitud de hora.
4. El proceso de servidor de hora responde con la hora actual.
5. Después se libera la conexión de transporte.

Observe que en el *host* 2 podría haber otros servidores enlazados a otros TSAPs en espera de conexiones entrantes sobre el mismo NSAP.

El panorama que hemos bosquejado está muy bien, excepto que hemos ocultado un pequeño problema: ¿cómo sabe el proceso de usuario del *host* 1 que el servidor de hora del día está conectado al TSAP 1522? Una posibilidad es que el servidor de hora del día se ha estado conectando al TSAP 1522 durante años, y gradualmente todos los usuarios de la red han aprendido esto. En este modelo, los servicios tienen direcciones TSAP estables que se listan en archivos en lugares bien conocidos, como el archivo */etc/services* de los sistemas UNIX, que lista cuáles servidores están enlazados de manera permanente a cuáles puertos.

Aunque las direcciones TSAP estables podrían funcionar bien con una cantidad pequeña de servicios clave que nunca cambian (por ejemplo, el servidor Web), en general, los procesos de usuario frecuentemente desean comunicarse con otros procesos de usuario que sólo existen durante un tiempo corto y no tienen una dirección TSAP conocida por adelantado. Es más, si puede haber muchos procesos de servidor, la mayoría de los cuales se usan pocas veces, es un desperdicio tenerlos activados a todos, escuchando en una dirección TSAP estable todo el día. En pocas palabras, se requiere un mejor esquema.

En la figura 6-9 se muestra un esquema simplificado. Se conoce como **protocolo inicial de conexión.** En lugar de que cada servidor concebible escuche en un TSAP bien conocido, cada máquina que desea ofrecer servicio a usuarios remotos tiene un **servidor de procesos** especial que actúa como *proxy* de los servidores de menor uso. Este servidor escucha en un grupo de puertos al mismo tiempo, esperando una solicitud de conexión. Los usuarios potenciales de un servicio comienzan por emitir una solicitud CONNECT, especificando la dirección TSAP del servicio que desean. Si no hay ningún servidor esperándolos, consiguen una conexión al servidor de procesos, como se muestra en la figura 6-9(a).

Tras obtener la solicitud entrante, el servidor de procesos genera el servidor solicitado, permitiéndole heredar la conexión con el usuario existente. El nuevo servidor entonces hace el trabajo requerido, mientras que el servidor de procesos retorna a escuchar solicitudes nuevas, como se muestra en la figura 6-9(b).

Aunque el protocolo de conexión inicial funciona bien para aquellos servidores que pueden crearse conforme son necesarios, hay muchas situaciones en las que los servicios existen independientemente del servidor de procesos. Por ejemplo, un servidor de archivos necesita operar en un *hardware* especial (una máquina con un disco) y no puede simplemente crearse sobre la marcha cuando alguien quiere comunicarse con él.

Para manejar esta situación, se usa con frecuencia un esquema alterno. En este modelo, existe un proceso especial llamado **servidor de nombres,** o a veces **servidor de directorio.** Para encontrar la dirección TSAP correspondiente a un nombre de servicio dado, como "hora del día", el usuario establece una conexión con el servidor de nombres (que escucha en un TSAP bien

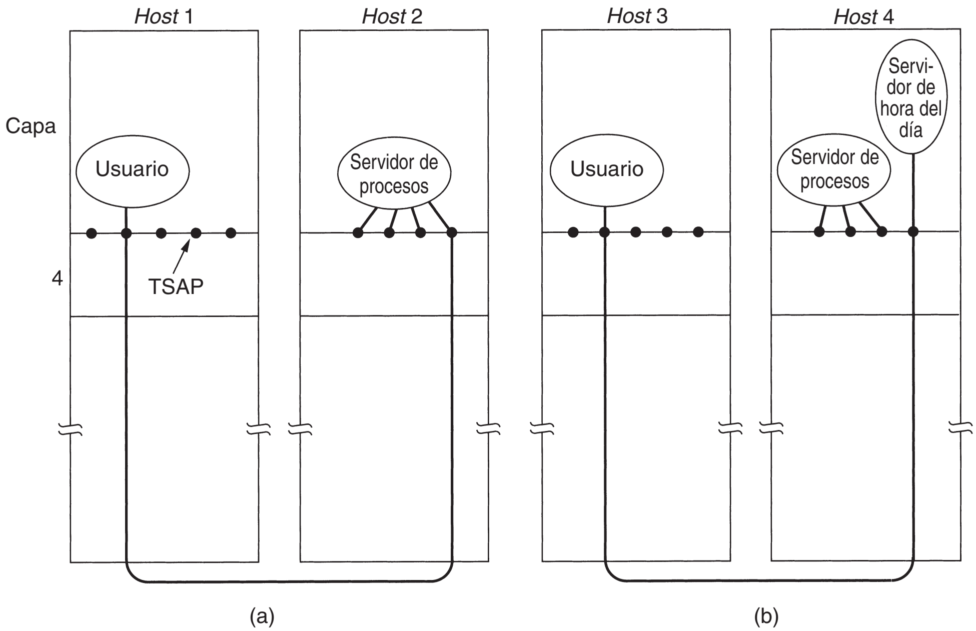


Figura 6-9. Manera en que un proceso de usuario del *host* 1 establece una conexión con un servidor de hora del día del *host* 2.

conocido). Entonces el usuario envía un mensaje especificando el nombre del servicio, y el servidor de nombres devuelve la dirección TSAP. Luego el usuario libera la conexión con el servidor de nombres y establece una nueva con el servicio deseado.

En este modelo, al crearse un servicio nuevo, debe registrarse en el servidor de nombres, dando tanto su nombre de servicio (generalmente, una cadena ASCII) como la dirección de su TSAP. El servidor de nombres registra esta información en su base de datos interna por lo que, cuando llegan solicitudes posteriores, sabe las respuestas.

La función del servidor de nombres es análoga al operador de asistencia de directorio del sistema telefónico: proporciona el número correspondiente a un nombre determinado. Al igual que en el sistema telefónico, es esencial que la dirección bien conocida del TSAP usado por el servidor de nombres (o el servidor de procesos del protocolo de conexión inicial) en realidad sea bien conocida. Si usted no conoce el número del operador de información, no puede llamar al operador de información para averiguarlo. Si usted cree que el número que se marca para obtener información es obvio, inténtelo alguna vez en otro país.

6.2.2 Establecimiento de una conexión

El establecimiento de una conexión suena fácil, pero en realidad es sorprendentemente complicado. A primera vista, parecería suficiente con que una entidad de transporte enviara una TPDU

CONNECTION REQUEST al destino y esperar una respuesta CONNECTION ACCEPTED. El problema ocurre cuando la red puede perder, almacenar o duplicar paquetes. Este comportamiento causa complicaciones serias.

Imagine una subred que está tan congestionada que las confirmaciones de recepción casi nunca regresan a tiempo, y cada paquete expira y se retransmite dos o tres veces. Suponga que la subred usa datagramas internamente, y que cada paquete sigue una ruta diferente. Algunos de los paquetes podrían atorarse en un congestionamiento de tráfico dentro de la subred y tardar mucho tiempo en llegar, es decir, se almacenarían en la subred y reaparecerían mucho después.

La peor pesadilla posible es la que sigue. Un usuario establece una conexión con un banco, envía mensajes indicando al banco que transfiera una gran cantidad de dinero a la cuenta de una persona no del todo confiable y a continuación libera la conexión. Por mala fortuna, cada paquete de la transacción se duplica y almacena en la subred. Tras liberar la conexión, todos los paquetes salen de la subred y llegan al destino en orden, solicitando al banco que establezca una conexión nueva, transfiera el dinero (nuevamente) y libere la conexión. El banco no tiene manera de saber que son duplicados; debe suponer que ésta es una segunda transacción independiente, y transfiere nuevamente el dinero. Durante el resto de esta sección estudiaremos el problema de los duplicados retardados, haciendo hincapié en los algoritmos para establecer conexiones de una manera confiable, de modo que pesadillas como la anterior no puedan ocurrir.

El meollo del problema es la existencia de duplicados retrasados. Esto puede atacarse de varias maneras, ninguna de las cuales es muy satisfactoria. Una es usar direcciones de transporte desechables. En este enfoque, cada vez que se requiere una dirección de transporte, se genera una nueva. Al liberarse la conexión, se descarta la dirección y no se vuelve a utilizar. Esta estrategia imposibilita el modelo de servidor de procesos de la figura 6-9.

Otra posibilidad es dar a cada conexión un identificador de conexión (es decir, un número de secuencia que se incrementa con cada conexión establecida), seleccionado por la parte iniciadora, y ponerlo en cada TPDU, incluida la que solicita la conexión. Tras la liberación de una conexión, cada entidad de transporte podría actualizar una tabla que liste conexiones obsoletas como pares (entidad de transporte igual, identificador de conexión). Cada vez que entrara una solicitud de conexión, podría cotejarse con la tabla para saber si pertenece a una conexión previamente liberada.

Por desgracia, este esquema tiene una falla básica: requiere que cada entidad de transporte mantenga una cierta cantidad de información histórica durante un tiempo indefinido. Si se cae una máquina y pierde su memoria, ya no sabrá qué identificadores de conexión usó.

Más bien, necesitamos un enfoque diferente. En lugar de permitir que los paquetes vivan eternamente en la subred, debemos diseñar un mecanismo para eliminar a los paquetes viejos que aún andan vagando por ahí. Si podemos asegurar que ningún paquete viva más allá de cierto tiempo conocido, el problema se vuelve algo más manejable.

El tiempo de vida de un paquete puede restringirse a un máximo conocido usando una de las siguientes técnicas:

1. Un diseño de subred restringido.
2. Colocar un contador de saltos en cada paquete.
3. Marcar el tiempo en cada paquete.

El primer método incluye cualquier método que evite que los paquetes hagan ciclos, combinado con una manera de limitar el retardo por congestionamientos a través de la trayectoria más larga posible (ahora conocida). El segundo método consiste en inicializar el conteo de saltos con un valor apropiado y decrementarlo cada vez que se reenvíe el paquete. El protocolo de red simplemente descarta cualquier paquete cuyo contador de saltos llega a cero. El tercer método requiere que cada paquete lleve la hora en la que fue creado, y que los enrutadores se pongan de acuerdo en descartar cualquier paquete que haya rebasado cierto tiempo predeterminado. Este último método requiere que los relojes de los enrutadores estén sincronizados, lo que no es una tarea fácil a menos que se logre la sincronización externamente a la red, por ejemplo, utilizando GPS o alguna estación de radio que difunda la hora exacta periódicamente.

En la práctica, necesitaremos garantizar no sólo que el paquete está eliminado, sino que también lo están todas sus confirmaciones de recepción, por lo que ahora introduciremos *T*, que es un múltiplo pequeño del tiempo de vida de paquete máximo verdadero. El múltiplo depende del protocolo, y simplemente tiene el efecto de hacer más grande a *T*. Si esperamos un tiempo *T* tras el envío de un paquete, podemos estar seguros de que todos los rastros suyos ya han desaparecido, y que ni él ni sus confirmaciones de recepción aparecerán repentinamente de la nada para complicar el asunto.

Al limitar los tiempos de vida de los paquetes, es posible proponer una manera a prueba de errores de establecer conexiones seguras. El método descrito a continuación se debe a Tomlinson (1975); resuelve el problema pero presenta algunas peculiaridades propias. El método fue refinado por Sunshine y Dalal (1978). En la práctica se usan ampliamente variantes suyas, incluso en TCP.

Para resolver el problema de una máquina que pierde toda la memoria acerca de su estado tras una caída, Tomlinson propuso equipar cada *host* con un reloj de hora del día. Los relojes de los diferentes *hosts* no necesitan estar sincronizados. Se supone que cada reloj tiene la forma de un contador binario que se incrementa a sí mismo a intervalos uniformes. Además, la cantidad de bits del contador debe ser igual o mayor que la cantidad de bits en los números de secuencia. Por último, y lo más importante, se supone que el reloj continúa operando aun ante la caída del *host*.

La idea básica es asegurar que nunca estén pendientes al mismo tiempo dos TPDUs de número idéntico. Cuando se establece una conexión, los *k* bits de orden menor del reloj se usan como número inicial de secuencia (también *k* bits). Por tanto, y a diferencia de los protocolos del capítulo 3, cada conexión comienza a numerar sus TPDUs con un número de secuencia inicial diferente. El espacio de secuencia también debe ser lo bastante grande para que, al regresar al principio de los números de secuencia, las TPDUs viejas con el mismo número de secuencia hayan desaparecido hace mucho tiempo. En la figura 6-10 se muestra esta relación lineal entre tiempo y números secuenciales iniciales.

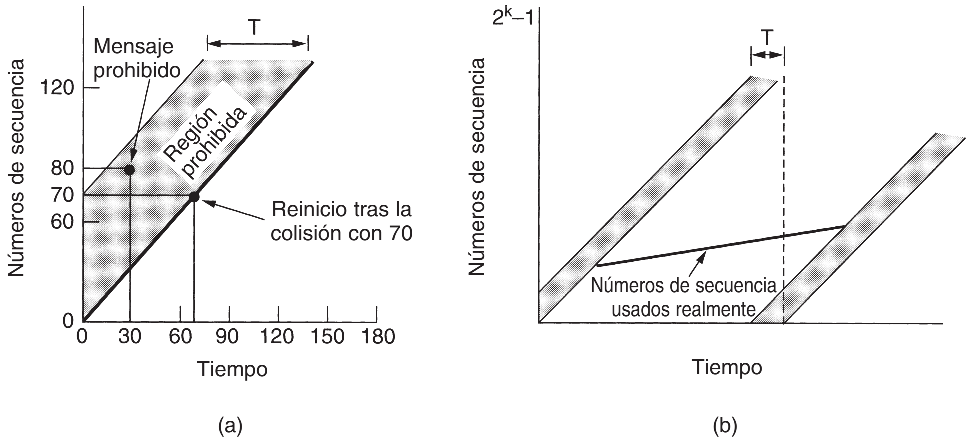


Figura 6-10. (a) Las TPDUs no pueden entrar en la zona prohibida. (b) El problema de la resincronización.

Una vez que ambas entidades de transporte han acordado el número de secuencia inicial, puede usarse cualquier protocolo de ventana corrediza para el control de flujo de datos. En realidad, la curva inicial de números de secuencia (indicada por la línea gruesa) no es realmente lineal, sino una escalera, ya que el reloj avanza en pasos discretos. Por sencillez, ignoraremos este detalle.

Cuando un *host* se cae ocurre un problema. Al reactivarse, sus entidades de transporte no saben dónde estaban en el espacio de secuencia. Una solución es requerir que las entidades de transporte estén inactivas durante *T* segundos tras una recuperación para permitir que todas las TPDUs viejas expiren. Sin embargo, en una interred compleja, *T* puede ser bastante grande, por lo que no es atractiva esta estrategia.

Para evitar requerir *T* seg de tiempo muerto tras una caída, es necesario introducir una nueva restricción en el uso de números de secuencia. Podemos ver claramente la necesidad de esta restricción mediante un ejemplo. Sea *T*, el tiempo máximo de vida de un paquete, 60 seg, y que el pulso del reloj sea de uno por segundo. Como lo indica la línea gruesa en la figura 6-10(a), el número de secuencia inicial de una conexión abierta en el momento *x* será *x*. Imagine que, en $t = 30$ seg, una TPDU de datos ordinaria enviada a través de la conexión 5 (previamente abierta), recibe el número de secuencia 80. Llamemos *X* a esta TPDU. De inmediato tras el envío de la TPDU *X*, el *host* se cae y reinicia pronto. En $t = 60$, el *host* comienza a reabrir las conexiones 0 a 4. En $t = 70$, reabre la conexión 5, usando el número de secuencia inicial 70, como se requiere. Durante los siguientes 15 segundos envía las TPDUs de datos 70 a 80. Por tanto, en $t = 85$ se ha inyectado una TPDU nueva con número de secuencia 80 y conexión 5 en la subred. Por desgracia, la TPDU *X* aún existe. Si ésta llegara al receptor antes de la nueva TPDU 80, la TPDU *X* sería aceptada y la TPDU 80 correcta sería rechazada como duplicado.

Para evitar tales problemas, debemos evitar la asignación de números de secuencia nuevos (es decir, asignados a TPDUs nuevas) durante un tiempo *T* antes de su uso potencial como números iniciales de secuencia. Las combinaciones ilegales de tiempo y número de secuencia se muestran

como la **región prohibida** en la figura 6-10(a). Antes de enviar cualquier TPDU por alguna conexión, la entidad de transporte debe leer el reloj y comprobar que no está en la región prohibida.

El protocolo puede meterse en problemas de dos maneras. Si un *host* envía demasiados datos con demasiada rapidez a través de una conexión recién abierta, la curva de número de secuencia real contra tiempo puede subir con mayor rapidez que la curva de número de secuencia inicial contra tiempo. Esto significa que la tasa de datos máxima en cualquier conexión es de una TPDU por pulso de reloj, y también significa que la entidad de transporte debe esperar hasta que el reloj pulse antes de abrir una nueva conexión tras un reinicio por cada caída, no sea que se use dos veces el mismo número de secuencia. Ambos puntos son argumentos a favor de un pulso de reloj corto (unos cuantos milisegundos).

Desgraciadamente, el ingreso en la región prohibida desde abajo al enviar con demasiada rapidez no es la única manera de meterse en problemas. Por la figura 6-10(b) debe quedar claro que con cualquier tasa de datos menor que la tasa del reloj, la curva de números de secuencia reales usados contra tiempo tarde o temprano entrará en la región prohibida por la izquierda. Cuanto mayor sea la pendiente de la curva de números de secuencia reales, mayor será el retardo de este evento. Como ya indicamos, justo antes de enviar cada TPDU, la entidad de transporte debe comprobar que no esté a punto de entrar en la región prohibida; de ser así, debe retardar la TPDU durante *T* seg o resincronizar los números de secuencia.

El método basado en reloj resuelve el problema del duplicado retrasado de las TPDUs de datos, pero para que este método resulte de utilidad, debe establecerse primero una conexión. Dado que las TPDUs de control también pueden retrasarse, hay el problema potencial de lograr que ambos lados acuerden el número de secuencia inicial. Supongamos, por ejemplo, que se establecen conexiones haciendo que el *host* 1 envíe una TPDU CONNECTION REQUEST con el número de secuencia inicial y el número de puerto de destino a un igual remoto, el *host* 2. El receptor, el *host* 2, confirma entonces la recepción de esta solicitud enviando de regreso una TPDU CONNECTION ACCEPTED. Si la TPDU CONNECTION REQUEST se pierde, pero aparece con retardo una CONNECTION REQUEST duplicada en el *host* 2, se establecerá incorrectamente la conexión.

Para resolver este problema, Tomlinson (1975) desarrolló el **acuerdo de tres vías** (*three-way handshake*). Este protocolo de establecimiento no requiere que ambos lados comiencen a transmitir con el mismo número de secuencia, por lo que puede usarse con otros métodos de sincronización distintos del método de reloj global. El procedimiento normal de establecimiento al iniciar el *host* 1 se muestra en la figura 6-11(a). El *host* 1 escoge un número de secuencia, *x*, y envía al *host* 2 una TPDU CONNECTION REQUEST que lo contiene. El *host* 2 responde con una TPDU CONNECTION ACCEPTED confirmando la recepción de *x* y anunciando su propio número de secuencia inicial, *y*. Por último, el *host* 1 confirma la recepción de la selección de un número de secuencia inicial del *host* 2 en la primera TPDU de datos que envía.

Ahora veamos la manera en que funciona el acuerdo de tres vías en presencia de TPDUs de control duplicadas con retraso. En la figura 6-11(b), la primera TPDU es una CONNECTION REQUEST duplicada con retraso de una conexión vieja. Esta TPDU llega al *host* 2 sin el conocimiento del *host* 1. El *host* 2 reacciona a esta TPDU enviando al *host* 1 una TPDU ACK, solicitando de hecho la comprobación de que el *host* 1 en verdad trató de establecer una nueva conexión. Al rechazar el *host* 1 el intento de establecimiento de conexión del *host* 2, éste se da cuenta de que fue

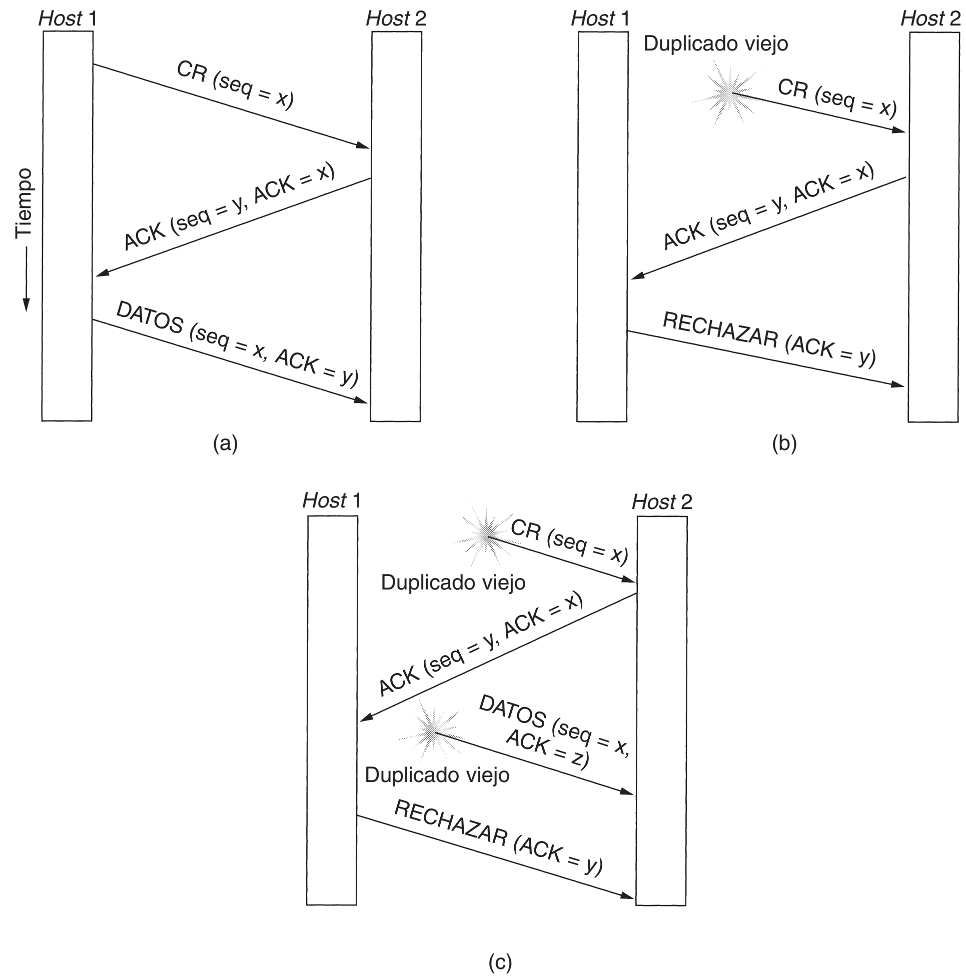


Figura 6-11. Tres escenarios para establecer una conexión usando un acuerdo de tres vías. CR significa CONNECTION REQUEST. (a) Operación normal. (b) CONNECTION REQUEST duplicada vieja que aparece de la nada. (c) CONNECTION REQUEST duplicada y ACK duplicada.

engañado por un duplicado con retardo y abandona la conexión. De esta manera, un duplicado con retardo no causa daño.

El peor caso ocurre cuando en la subred deambulan tanto una CONNECTION REQUEST retardada como una ACK. Este caso se muestra en la figura 6-11(c). Como en el ejemplo previo, el *host* 2 recibe una CONNECTION REQUEST retrasada y la contesta. En este momento es crucial notar que el *host* 2 ha propuesto usar *y* como número de secuencia inicial para el tráfico del *host* 2 al *host* 1, sabiendo bien que no existen todavía TPDUs que contengan el número de secuencia *y* ni confirmaciones de recepción de *y*. Cuando llega la segunda TPDU retrasada al *host* 2, el hecho

de que se confirmó la recepción de *z* en lugar de *y* indica al *host* 2 que éste también es un duplicado viejo. Lo importante que se debe tomar en cuenta aquí es que no haya combinación de viejas TPDUs que puedan causar la falla del protocolo y permitan el establecimiento de una conexión accidentalmente cuando nadie la quiere.

6.2.3 Liberación de una conexión

La liberación de una conexión es más fácil que su establecimiento. No obstante, hay más escollos de los que uno podría imaginar. Como mencionamos antes, hay dos estilos de terminación de una conexión: liberación asimétrica y liberación simétrica. La liberación asimétrica es la manera en que funciona el sistema telefónico: cuando una parte cuelga, se interrumpe la conexión. La liberación simétrica trata la conexión como dos conexiones unidireccionales distintas, y requiere que cada una se libere por separado.

La liberación asimétrica es abrupta y puede resultar en la pérdida de datos. Considere el escenario de la figura 6-12. Tras establecerse la conexión, el *host* 1 envía una TPDU que llega adecuadamente al *host* 2. Entonces el *host* 1 envía otra TPDU. Desgraciadamente, el *host* 2 emite una DISCONNECT antes de llegar la segunda TPDU. El resultado es que se libera la conexión y se pierden datos.

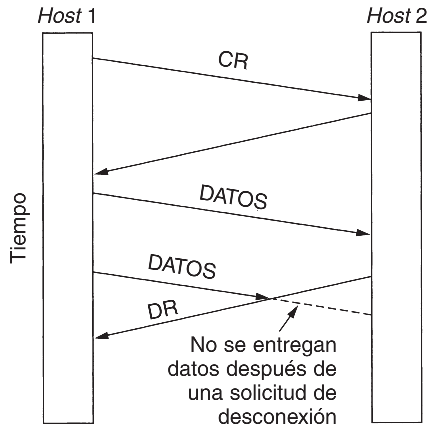


Figura 6-12. Desconexión abrupta con pérdida de datos.

Es obvio que se requiere un protocolo de liberación más refinado para evitar la pérdida de datos. Una posibilidad es usar la liberación simétrica, en la que cada dirección se libera independientemente de la otra. Aquí, un *host* puede continuar recibiendo datos aun tras haber enviado una TPDU DISCONNECT.

La liberación simétrica es ideal cuando cada proceso tiene una cantidad fija de datos por enviar y sabe con certidumbre cuándo los ha enviado. En otras situaciones, la determinación de si

se ha efectuado o no todo el trabajo y si debe terminarse o no la conexión no es tan obvia. Podríamos pensar en un protocolo en el que el *host* 1 diga: "Ya terminé. ¿Terminaste también?" Si el *host* 2 responde: "Ya terminé también. Adiós", la conexión puede liberarse con seguridad.

Por desgracia, este protocolo no siempre funciona. Hay un problema famoso que tiene que ver con ese asunto; se llama **problema de los dos ejércitos.** Imagine que un ejército blanco está acampado en un valle, como se muestra en la figura 6-13. En los dos cerros que rodean al valle hay ejércitos azules. El ejército blanco es más grande que cualquiera de los dos ejércitos azules por separado, pero juntos éstos son más grandes que el ejército blanco. Si cualquiera de los dos ejércitos azules ataca por su cuenta, será derrotado, pero si los dos atacan simultáneamente obtendrán la victoria.

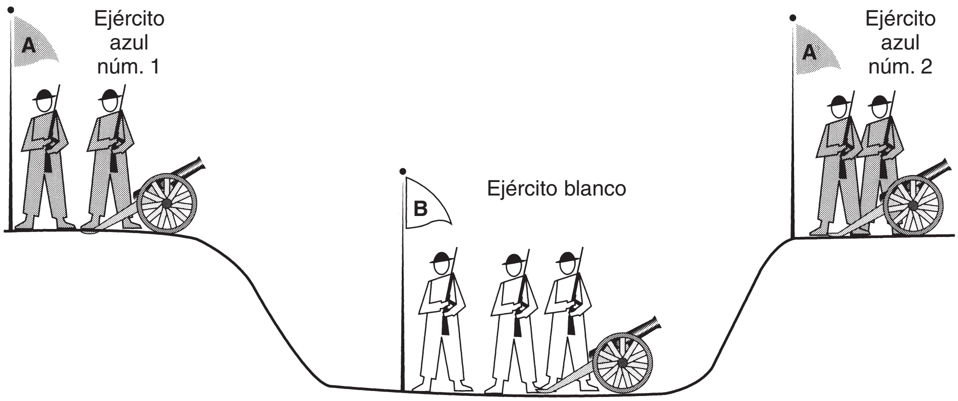


Figura 6-13. Problema de los dos ejércitos.

Los ejércitos azules quieren sincronizar sus ataques. Sin embargo, su único medio de comunicación es el envío de mensajeros a pie a través del valle, donde podrían ser capturados, perdiéndose el mensaje (es decir, tienen que usar un canal de comunicación no confiable). La pregunta es: ¿Existe un protocolo que permita que los ejércitos azules ganen?

Supongamos que el comandante del ejército azul núm. 1 envía un mensaje que dice: "Propongo que ataquemos al amanecer del 29 marzo. ¿Qué les parece?" Ahora supongamos que llega el mensaje y que el comandante del ejército azul núm. 2 está de acuerdo, y que su respuesta llega con seguridad al ejército azul núm. 1. ¿Ocurrirá el ataque? Probablemente no, porque el comandante núm. 2 no sabe si su respuesta llegó. Si no llegó, el ejército azul núm. 1 no atacará, y sería tonto de su parte emprender el ataque.

Mejoremos ahora el protocolo haciéndolo un acuerdo de tres vías. El iniciador de la propuesta original debe confirmar la recepción de la respuesta. Suponiendo que no se pierden mensajes, el ejército azul núm. 2 recibirá la confirmación de recepción, pero ahora el que dudará será el comandante del ejército azul núm. 1. A fin de cuentas, no sabe si ha llegado su confirmación de

recepción y, si no llegó, sabe que el ejército núm. 2 no atacará. Podríamos probar ahora un protocolo de acuerdo de cuatro vías, pero tampoco ayudaría.

De hecho, puede demostrarse que no existe un protocolo que funcione. Supongamos que existiera algún protocolo. O el último mensaje del protocolo es esencial, o no lo es. Si no lo es, hay que eliminarlo (así como los demás mensajes no esenciales) hasta que quede un protocolo en el que todos los mensajes sean esenciales. ¿Qué ocurre si el mensaje final no pasa? Acabamos de decir que es esencial, por lo que, si se pierde, el ataque no ocurrirá. Dado que el emisor del mensaje final nunca puede estar seguro de su llegada, no se arriesgará a atacar. Peor aún, el otro ejército azul sabe esto, por lo que no atacará tampoco.

Para ver la aplicación del problema de los dos ejércitos a la liberación de conexiones, simplemente sustituya "atacar" por "desconectar". Si ninguna de las partes está preparada para desconectarse hasta estar convencida de que la otra está preparada para desconectarse también, nunca ocurrirá la desconexión.

En la práctica, generalmente estamos más dispuestos a correr riesgos al liberar conexiones que al atacar ejércitos blancos, por lo que la situación no es del todo desesperada. En la figura 6-14 se ilustran cuatro escenarios de liberación usando un acuerdo de tres vías. Aunque este protocolo no es infalible, generalmente es adecuado.

En la figura 6-14(a) vemos el caso normal en el que uno de los usuarios envía una TPDU DR (DISCONNECTION REQUEST, solicitud de desconexión) a fin de iniciar la liberación de una conexión. Al llegar, el receptor devuelve también una TPDU DR e inicia un temporizador, para el caso de que se pierda su DR. Al llegar esta DR, el emisor original envía de regreso una TPDU ACK y libera la conexión. Finalmente, cuando la TPDU ACK llega, el receptor también libera la conexión. La liberación de una conexión significa que la entidad de transporte remueve la información sobre la conexión de su tabla de conexiones abiertas y avisa de alguna manera al dueño de la conexión (el usuario de transporte). Esta acción es diferente a aquélla en la que el usuario de transporte emite una primitiva DISCONNECT.

Si se pierde la última TPDU ACK, como se muestra en la figura 6-14(b), el temporizador salva la situación. Al expirar el temporizador, la conexión se libera de todos modos.

Ahora consideremos el caso de la pérdida de la segunda DR. El usuario que inicia la desconexión no recibirá la respuesta esperada, su temporizador expirará y todo comenzará de nuevo. En la figura 6-14(c) vemos la manera en que funciona esto, suponiendo que la segunda vez no se pierden TPDUs y que todas se entregan correctamente y a tiempo.

Nuestro último escenario, la figura 6-14(d), es el misma que en la figura 6-14(c), excepto que ahora suponemos que todos los intentos repetidos de retransmitir la DR también fallan debido a la pérdida de TPDUs. Tras *N* reintentos, el emisor simplemente se da por vencido y libera la conexión. Mientras tanto, expira el temporizador del receptor y también se sale.

Aunque este protocolo generalmente es suficiente, en teoría puede fallar si se pierden la DR inicial y *N* retransmisiones. El emisor se dará por vencido y liberará la conexión, pero el otro lado no sabrá nada sobre los intentos de desconexión y seguirá plenamente activo. Esta situación origina una conexión semiabierta.

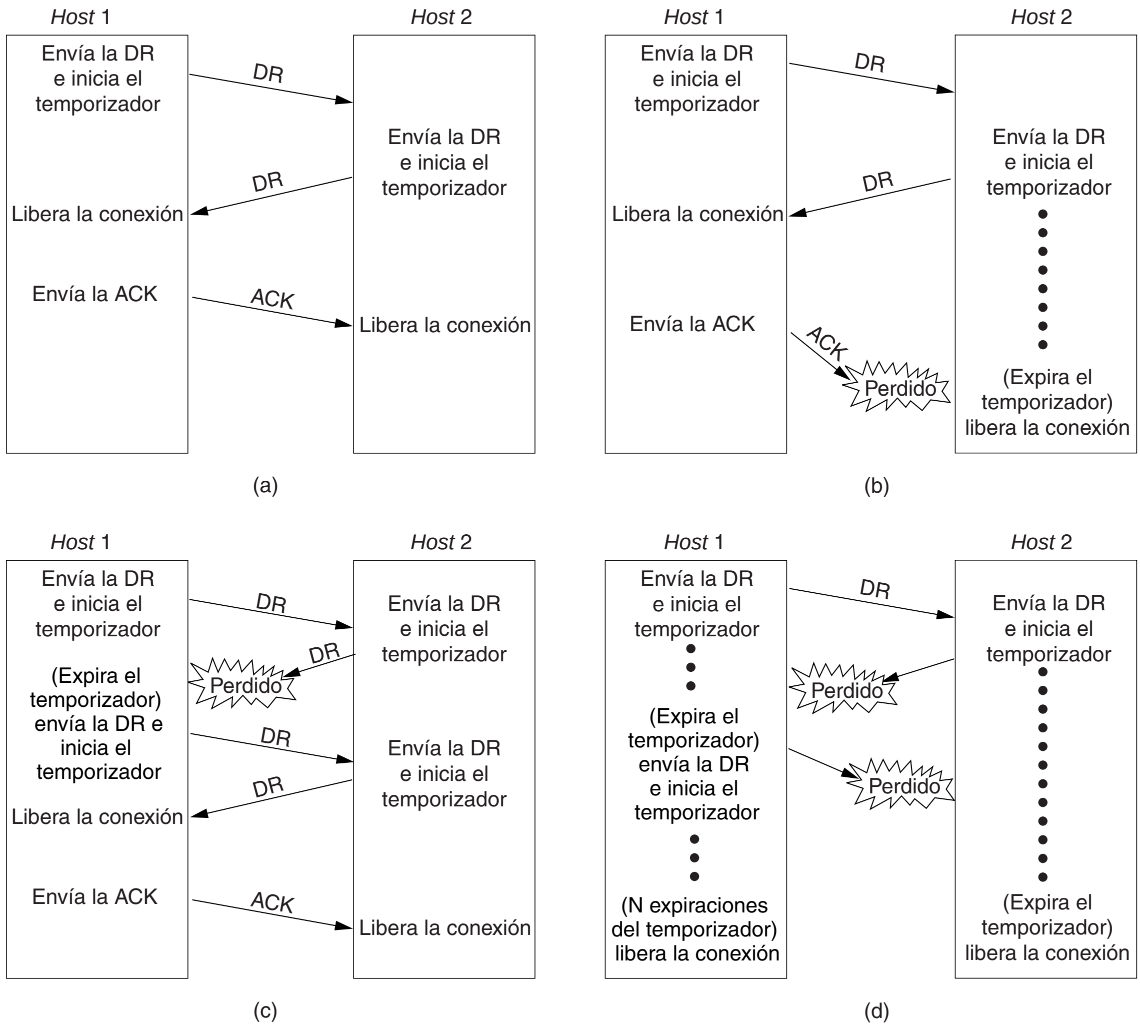


Figura 6-14. Cuatro escenarios de un protocolo para liberar una conexión. (a) Caso normal del acuerdo de tres vías. (b) Pérdida de la última ACK. (c) Respuesta perdida. (d) Respuesta perdida y pérdida de las DRs subsecuentes.

Pudimos haber evitado este problema no permitiendo que el emisor se diera por vencido tras *N* reintentos, sino obligándolo a seguir insistiendo hasta recibir una respuesta. Sin embargo, si se permite que expire el temporizador en el otro lado, entonces el emisor continuará eternamente, pues nunca aparecerá una respuesta. Si no permitimos que expire el temporizador en el lado receptor, entonces el protocolo se atora en la figura 6-14(b).

Otra manera de eliminar las conexiones semiabiertas es tener una regla que diga que, si no ha llegado ninguna TPDU durante cierta cantidad de segundos, se libera automáticamente la conexión. De esta manera, si un lado llega a desconectarse, el otro lado detectará la falta de actividad

y también se desconectará. Por supuesto que si se pone en práctica esta regla, es necesario que cada entidad de transporte tenga un temporizador que se detenga y se reinicie con cada envío de una TPDU. Si expira este temporizador, se transmite una TPDU ficticia, simplemente para evitar que el otro lado se desconecte. Por otra parte, si se usa la regla de desconexión automática y se pierden demasiadas TPDU ficticias una tras otra en una conexión que de otro modo estaría en reposo, primero un lado y luego el otro se desconectarán automáticamente.

No insistiremos más en este punto, pero ya debe quedar claro que la liberación de una conexión no es ni remotamente tan sencilla como parece inicialmente.

6.2.4 Control de flujo y almacenamiento en búfer

Habiendo examinado el establecimiento y la liberación de conexiones con algún detalle, veamos ahora la manera en que se manejan las conexiones mientras están en uso. Ya conocemos uno de los aspectos clave: el control de flujo. En algunos sentidos el problema del control de flujo en la capa de transporte es igual que en la capa de enlace de datos, pero en otros es diferente. La similitud básica es que en ambas capas se requiere una ventana corrediza u otro esquema en cada conexión para evitar que un emisor rápido desborde a un receptor lento. La diferencia principal es que un enrutador por lo regular tiene relativamente pocas líneas, y que un *host* puede tener numerosas conexiones. Esta diferencia hace que sea impráctico implementar en la capa de transporte la estrategia de almacenamiento en búfer de la capa de enlace de datos.

En los protocolos de enlace de datos del capítulo 3, las tramas se almacenaban en búferes tanto en el enrutador emisor como en el receptor. Por ejemplo, en el protocolo 6 se requiere que tanto el emisor como el receptor dediquen *MAX_SEQ* + 1 búferes a cada línea, la mitad para entradas y la mitad para salidas. En un *host* con un máximo de, digamos, 64 conexiones y un número de secuencia de 4 bits, este protocolo requerirá 1024 búferes.

En la capa de enlace de datos, el lado emisor debe almacenar en búfer las tramas de salida porque cabe la posibilidad de que tengan que retransmitirse. Si la subred provee un servicio de datagramas, la entidad de transporte emisora también debe manejar búferes, por la misma razón. Si el receptor sabe que el emisor almacena en búfer todas las TPDUs hasta que se confirma su recepción, el receptor podría o no dedicar búferes específicos a conexiones específicas, según considere necesario. Por ejemplo, el receptor podría mantener un solo grupo de búferes compartido por todas las conexiones. Cuando entra una TPDU, se hace un intento por adquirir dinámicamente un búfer nuevo. Si hay uno disponible, se acepta la TPDU; de otro modo, se descarta. Dado que el emisor está preparado para retransmitir las TPDUs perdidas por la subred, no hay nada de malo en hacer que el receptor se deshaga de las TPDUs, aunque se desperdicien algunos recursos. El emisor simplemente sigue intentando hasta que recibe una confirmación de recepción.

En resumen, si el servicio de red no es confiable, el emisor debe almacenar en búfer todas las TPDUs enviadas, igual que en la capa de enlace de datos. Sin embargo, con un servicio de red confiable son posibles otros arreglos. En particular, si el emisor sabe que el receptor siempre tiene espacio de búfer, no necesita retener copias de las TPDUs que envía. Sin embargo, si el receptor no puede garantizar que se aceptará cada TPDU que llegue, el emisor tendrá que usar búferes de

todas maneras. En el último caso, el emisor no puede confiar en la confirmación de recepción de la capa de red porque esto sólo significa que ha llegado la TPDU, no que ha sido aceptada. Regresaremos después a este importante punto.

Aun si el receptor está de acuerdo en utilizar búferes, todavía queda la cuestión del tamaño de éstos. Si la mayoría de las TPDUs tiene aproximadamente el mismo tamaño, es natural organizar los búferes como un grupo de búferes de tamaño idéntico, con una TPDU por búfer, como en la figura 6-15(a). Sin embargo, si hay una variación grande en el tamaño de las TPDUs, de unos cuantos caracteres ingresados en una terminal hasta miles de caracteres provenientes de transferencias de archivos, el grupo de búferes de tamaño fijo presenta problemas. Si el tamaño de búfer se escoge igual a la TPDU más grande, se desperdiciará espacio cada vez que llegue una TPDU corta. Si el tamaño se escoge menor que el tamaño máximo de TPDU, se requerirán varios búferes para las TPDUs grandes, con la complejidad inherente.

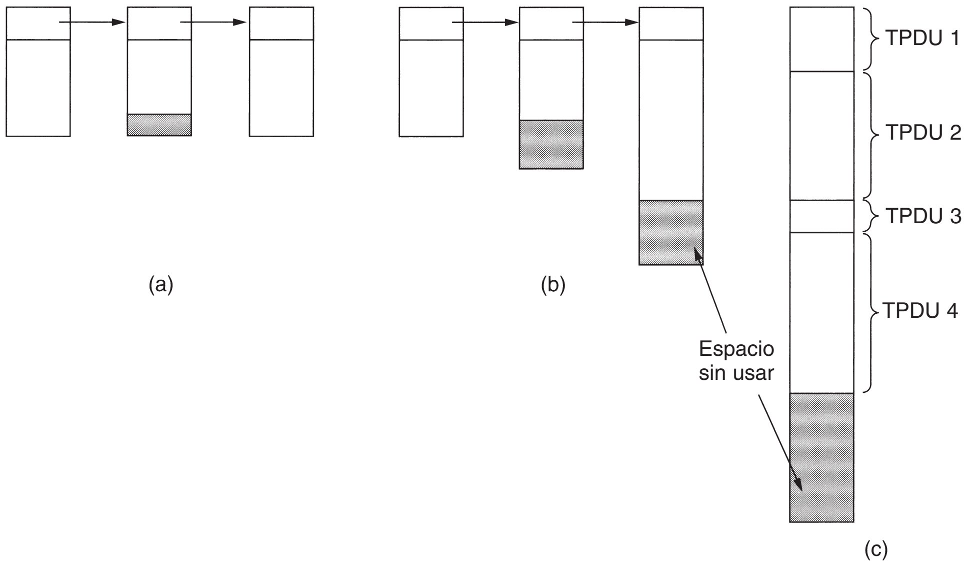


Figura 6-15. (a) Búferes encadenados de tamaño fijo. (b) Búferes encadenados de tamaño variable. (c) Un gran búfer circular por conexión.

Otra forma de enfrentar el problema del tamaño de los búferes es el uso de búferes de tamaño variable, como en la figura 6-15(b). La ventaja aquí es un mejor uso de la memoria, al costo de una administración de búferes más complicada. Una tercera posibilidad es dedicar un solo búfer circular grande por conexión, como en la figura 6-15(c). Este sistema también hace buen uso de la memoria cuando todas las conexiones tienen una carga alta, pero es deficiente si algunas conexiones cuentan con poca carga.

El equilibrio óptimo entre el almacenamiento en búfer en el origen y en el destino depende del tipo de tráfico transportado por la conexión. Para un tráfico de bajo ancho de banda con ráfagas, como el producido por una terminal interactiva, es mejor no dedicarle búferes, sino adquirirlos de

manera dinámica en ambos extremos. Dado que el emisor no puede estar seguro de que el receptor será capaz de adquirir un búfer, el emisor debe retener una copia de la TPDU hasta recibir su confirmación de recepción. Por otra parte, para la transferencia de archivos y otro tráfico de alto ancho de banda, es mejor si el receptor dedica una ventana completa de búferes, para permitir el flujo de datos a máxima velocidad. Por tanto, para un tráfico en ráfagas de bajo ancho de banda, es mejor mantener búferes en el emisor; para tráfico continuo de alto ancho de banda, es mejor hacerlo en el receptor.

A medida que se abren y cierran conexiones, y a medida que cambia el patrón del tráfico, el emisor y el receptor necesitan ajustar dinámicamente sus asignaciones de búferes. En consecuencia, el protocolo de transporte debe permitir que un *host* emisor solicite espacio en búfer en el otro extremo. Los búferes podrían repartirse por conexión o, en conjunto, para todas las conexiones en operación entre los dos *hosts*. Como alternativa, el receptor, sabiendo su capacidad de manejo de búferes (pero sin saber el tráfico generado) podría indicar al emisor "te he reservado *X* búferes". Si aumentara la cantidad de conexiones abiertas, podría ser necesario reducir una asignación, por lo que el protocolo debe contemplar esta posibilidad.

Una manera razonablemente general de manejar la asignación dinámica de búferes es desacoplarlos de las confirmaciones de recepción, en contraste con los protocolos de ventana corrediza del capítulo 3. La administración dinámica de búferes implica, de hecho, una ventana de tamaño variable. Inicialmente, el emisor solicita una cierta cantidad de búferes, con base en sus necesidades percibidas. El receptor entonces otorga tantos búferes como puede. Cada vez que el emisor envía una TPDU, debe disminuir su asignación, deteniéndose por completo al llegar la asignación a cero. El receptor entonces incorpora tanto las confirmaciones de recepción como las asignaciones de búfer al tráfico de regreso.

En la figura 6-16 se muestra un ejemplo de la manera en que podría trabajar la administración dinámica de ventanas en una subred de datagramas con números de secuencia de 4 bits. Supongamos que la información de asignación de búferes viaja en TPDUs distintas, como se muestra, y no se incorpora en el tráfico de regreso. Inicialmente, *A* quiere ocho búferes, pero se le otorgan solamente cuatro. Entonces envía tres TPDUs, de las cuales se pierde la tercera. La TPDU 6 confirma la recepción de todas las TPDUs hasta el número de secuencia 1, inclusive, permitiendo por tanto que *A* libere esos búferes, y además informa a *A* que tiene permiso de enviar tres TPDUs más comenzando después de 1 (es decir, las TPDUs 2, 3 y 4). *A* sabe que ya ha enviado el número 2, por lo que piensa que debe enviar las TPDUs 3 y 4, lo que procede a hacer. En este punto se bloquea y debe esperar una nueva asignación de búfer. Por otro lado, las retransmisiones inducidas por expiraciones del temporizador (línea 9) sí pueden ocurrir durante el bloqueo, pues usan búferes ya asignados. En la línea 10, *B* confirma la recepción de todas las TPDUs hasta la 4, inclusive, pero se niega a permitir que *A* continúe. Tal situación es imposible con los protocolos de ventana fija del capítulo 3. La siguiente TPDU de *B* a *A* asigna otro búfer y permite a *A* continuar.

Pueden surgir problemas potenciales con los esquemas de asignación de búferes de este tipo en las redes de datagramas si hay pérdidas de TPDU. Observe la línea 16. *B* ha asignado ahora más búferes a *A*, pero la TPDU de asignación se perdió. Dado que las TPDUs de control no están en secuencia, *A* se encuentra estancado. Para evitar esta situación, cada *host* debe enviar periódica-

	A	Mensaje	B	Comentarios
1	→	<solicito 8 búferes>	→	A quiere 8 búferes
2	←	<ack = 15, buf = 4>	←	B sólo otorga los mensajes 0 a 3
3	→	<seq = 0, data = m0>	→	A tiene 3 búferes libres
4	→	<seq = 1, data = m1>	→	A tiene 2 búferes libres
5	→	<seq = 2, data = m2>	• • •	Mensaje perdido, pero A piensa que le queda 1
6	←	<ack = 1, buf = 3>	←	B confirma la recepción de 0 y 1, permite 2-4
7	→	<seq = 3, data = m3>	→	A tiene un búfer libre
8	→	<seq = 4, data = m4>	→	A tiene 0 búferes libres y debe detenerse
9	→	<seq = 2, data = m2>	→	El temporizador de A expira y retransmite
10	←	<ack = 4, buf = 0>	←	Todo confirmado, pero A bloqueado aún
11	←	<ack = 4, buf = 1>	←	A puede enviar 5 ahora
12	←	<ack = 4, buf = 2>	←	B encontró un búfer nuevo en algún lado
13	→	<seq = 5, data = m5>	→	A tiene 1 búfer libre
14	→	<seq = 6, data = m6>	→	A está bloqueado nuevamente
15	←	<ack = 6, buf = 0>	←	A aún está bloqueado
16	• • •	<ack = 6, buf = 4>	←	Bloqueo irreversible potencial

Figura 6-16. Asignación dinámica de búferes. Las flechas muestran la dirección de la transmisión. Los puntos suspensivos (...) indican una TPDU perdida.

mente una TPDU de control con la confirmación de recepción y estado de búferes de cada conexión. De esta manera, el estancamiento se romperá tarde o temprano.

Hasta ahora hemos supuesto tácitamente que el único límite impuesto a la tasa de datos del emisor es la cantidad de espacio de búfer disponible en el receptor. A medida que siga cayendo significativamente el precio de la memoria, podría llegar a ser factible equipar a los *hosts* con tanta memoria que la falta de búferes dejaría de ser un problema.

Si el espacio de búfer ya no limita el flujo máximo, aparecerá otro cuello de botella: la capacidad de carga de la subred. Si enrutadores adyacentes pueden intercambiar cuando mucho x paquetes/seg y hay k trayectorias sin traslape entre un par de *hosts*, no hay manera de que esos *hosts* puedan intercambiar más de kx TPDUs/seg, sin importar la cantidad de espacio de búfer disponible en cada terminal. Si el emisor presiona demasiado (es decir, envía más de kx TPDUs/seg), la subred se congestionará, pues será incapaz de entregar las TPDUs a la velocidad con que llegan.

Lo que se necesita es un mecanismo basado en la capacidad de carga de la subred en lugar de la capacidad de almacenamiento en búfer del receptor. Es claro que el mecanismo de control de flujo debe aplicarse al emisor para evitar que estén pendientes demasiadas TPDUs sin confirmación de recepción al mismo tiempo. Belsnes (1975) propuso el uso de un esquema de control de flujo de ventana corrediza en el que el emisor ajusta dinámicamente el tamaño de la ventana para igualarla a la capacidad de carga de la red. Si la red puede manejar c TPDUs/seg y el tiempo de ciclo (incluidos transmisión, propagación, encolamiento, procesamiento en el receptor y devolución de la confirmación de recepción) es de r, entonces la ventana del emisor debe ser cr. Con una ventana de este tamaño, el emisor normalmente opera con el canal a su máxima capacidad. Cualquier pequeña disminución en el desempeño de la red causará que se bloquee.

A fin de ajustar periódicamente el tamaño de la ventana, el emisor podría vigilar ambos parámetros y calcular después el tamaño de ventana deseado. La capacidad de carga puede determinarse con sólo contar la cantidad de TPDUs confirmadas durante algún periodo y dividirla entre el periodo. Durante la medición, el emisor debe enviar a la mayor velocidad posible, para asegurarse que sea la capacidad de carga de la red, y no la baja tasa de entrada, el factor limitante de la tasa de confirmaciones de recepción. El tiempo requerido para la confirmación de recepción de una TPDU transmitida puede medirse exactamente y mantenerse una media de operación. Dado que la capacidad de la red depende de la cantidad de tráfico en ella, debe ajustarse el tamaño de la ventana con frecuencia para responder a los cambios en la capacidad de carga. Como veremos después, Internet usa un esquema parecido.

6.2.5 Multiplexión

La multiplexión de varias conversaciones en conexiones, circuitos virtuales y enlaces físicos desempeña un papel importante en diferentes capas de la arquitectura de la red. En la capa de transporte puede surgir la necesidad de multiplexión por varias razones. Por ejemplo, si en un *host* sólo se dispone de una dirección de red, todas las conexiones de transporte de esa máquina tendrán que utilizarla. Cuando llega una TDPU, se necesita algún mecanismo para saber a cuál proceso asignarla. Esta situación, conocida como **multiplexión hacia arriba,** se muestra en la figura 6-17(a). En esta figura, cuatro distintas conexiones de transporte utilizan la misma conexión de red (por ejemplo, dirección IP) al *host* remoto.

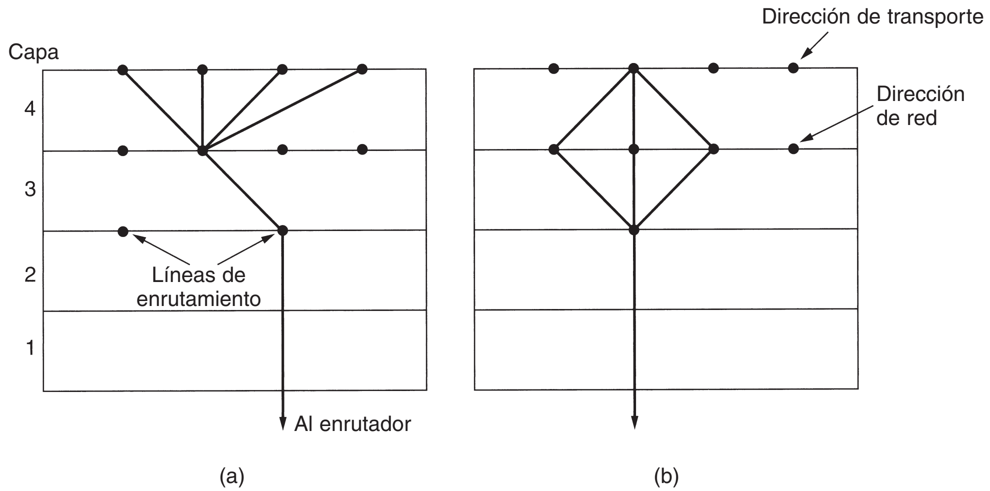


Figura 6-17. (a) Multiplexión hacia arriba. (b) Multiplexión hacia abajo.

La multiplexión también puede ser útil en la capa de transporte por otra razón. Por ejemplo, supongamos que una subred utiliza circuitos virtuales de manera interna y que impone una tasa máxima de datos a cada uno. Si un usuario necesita más ancho de banda del que le puede proporcionar un circuito virtual, una alternativa es abrir múltiples conexiones de red y distribuir el tráfico entre ellas en *round robin* (de manera circular), como se muestra en la figura 6-17(b). Esto se denomina **multiplexión hacia abajo.** Con *k* conexiones de red abiertas, el ancho de banda efectivo se incrementa por un factor de *k.* Un ejemplo común de multiplexión hacia abajo se da en los usuarios caseros que poseen una línea ISDN. Esta línea proporciona dos conexiones separadas de 64 kbps cada una. Al usar ambas para llamar a un proveedor de Internet y dividir el tráfico en las dos líneas se consigue un ancho de banda efectivo de 128 kbps.

6.2.6 Recuperación de caídas

Si los *hosts* y los enrutadores están sujetos a caídas, la recuperación de éstas se vuelve un tema importante. Si la entidad de transporte está por entero dentro de los *hosts*, la recuperación de caídas de la red y de enrutadores es sencilla. Si la capa de red proporciona servicio de datagramas, las entidades de transporte esperan la pérdida de algunas TPDUs todo el tiempo, y saben cómo manejarla. Si la capa de red proporciona servicio orientado a la conexión, entonces la pérdida de un circuito virtual se maneja estableciendo uno nuevo y sondeando la entidad de transporte remota para saber cuáles TPDUs ha recibido y cuáles no. Estas últimas pueden retransmitirse.

Un problema más complicado es la manera de recuperarse de caídas del *host*. En particular, podría ser deseable que los clientes sean capaces de continuar trabajando cuando los servidores caen y se reinician rápidamente. Para ilustrar la dificultad, supongamos que un *host*, el cliente, envía un archivo grande a otro *host*, el servidor de archivos, usando un protocolo de parada y espera sencillo. La capa de transporte del servidor simplemente pasa las TPDUs entrantes al usuario de transporte, una por una. A medio camino de la transmisión se cae el servidor. Al reactivarse, sus tablas se reinicializan, por lo que ya no sabe con precisión dónde estaba.

En un intento por recuperar su estado previo, el servidor podría enviar una TPDU de difusión a todos los demás *hosts*, anunciando que acaba de caerse y solicitando a sus clientes que le informen el estado de todas las conexiones abiertas. Cada cliente puede estar en uno de dos estados: una TPDU pendiente, *S1*, o ninguna TPDU pendiente, *S0*. Con base en esta información de estado, el cliente debe decidir si retransmitirá o no la TPDU más reciente.

A primera vista parecería obvio: el cliente debe retransmitir sólo si tiene una TPDU pendiente sin confirmación de recepción (es decir, si está en el estado *S1*) cuando se entera de la caída. Sin embargo, una inspección más cercana revela dificultades con este enfoque ingenuo. Considere, por ejemplo, la situación en la que la entidad de transporte del servidor envía primero una confirmación de recepción y luego escribe en el proceso de aplicación. La escritura de una TPDU en este flujo de salida y el envío de una confirmación de recepción son dos eventos diferentes que no pueden hacerse simultáneamente. Si ocurre una caída tras el envío de la confirmación de recepción

pero antes de hacer la escritura, el cliente recibirá la confirmación de recepción y estará, por tanto, en el estado *S0* al llegar el anuncio de recuperación de la caída. Entonces el cliente no retransmitirá, pensando (incorrectamente) que la TPDU llegó. Esta decisión del cliente conduce a una TPDU faltante.

En este punto el lector podría pensar: "Ese problema se resuelve fácilmente. Todo lo que hay que hacer es reprogramar la entidad de transporte para que primero haga la escritura y luego envíe la confirmación de recepción". Piénselo mejor. Imagine que se ha hecho la escritura pero que la caída ocurre antes de poder enviar la confirmación de recepción. El cliente estará en el estado *S1*, y por tanto retransmitirá, conduciendo a una TPDU duplicada no detectada en el flujo de salida al proceso de aplicación del servidor.

Sin importar cómo se programen el emisor y el receptor, siempre habrá situaciones en las que el protocolo no podrá recuperarse correctamente. El servidor puede programarse de una de dos maneras: mandar confirmación de recepción primero o escribir primero. El cliente puede programarse de cuatro maneras: siempre retransmitir la última TPDU, nunca retransmitir la última TPDU, retransmitir sólo en el estado *S0* o retransmitir sólo en el estado *S1*. Esto da ocho combinaciones pero, como veremos, para cada combinación existe algún grupo de eventos que hacen fallar al protocolo.

Son posibles tres eventos en el servidor: el envío de una confirmación de recepción (*A*), la escritura al proceso de salida (*W*) y una caída (*C*). Los tres eventos pueden ocurrir en seis órdenes diferentes: *AC*(*W*), *AWC*, *C*(*AW*), *C*(*WA*), *WAC* y *WC*(*A*), donde los paréntesis se usan para indicar que ni *A* ni *W* pueden seguir a *C* (es decir, una vez que el servidor se cae, se cayó). En la figura 6-18 se muestran las ocho combinaciones de la estrategia cliente-servidor y las secuencias de eventos válidas para cada una. Observe que para cada estrategia hay alguna secuencia de eventos que causa la falla del protocolo. Por ejemplo, si el cliente siempre retransmite, el evento *AWC* generará un duplicado no detectado, aun si los otros dos eventos funcionan adecuadamente.

Hacer más elaborado el protocolo no sirve de nada. Aunque el cliente y el servidor intercambien varias TPDUs antes de que el servidor intente escribir, para que el cliente sepa exactamente lo que está a punto de ocurrir, el cliente no tiene manera de saber si ha ocurrido una caída justo antes o justo después de la escritura. La conclusión es inevitable: según nuestra regla básica de que no debe haber eventos simultáneos, la caída de un *host* y su recuperación no pueden hacerse transparentes a las capas superiores.

En términos más generales, este resultado puede replantearse como que la recuperación de una caída de capa *N* sólo puede hacerla la capa *N* + 1, y sólo si la capa superior retiene suficiente información del estado. Como se mencionó antes, la capa de transporte puede recuperarse de fallas de la capa de red, siempre y cuando cada extremo de una conexión lleve el registro de dónde está.

Este problema nos lleva al asunto de qué significa en realidad una confirmación de recepción de extremo a extremo. En principio, el protocolo de transporte es de extremo a extremo, y no está encadenado como las capas inferiores. Ahora considere el caso de un usuario que introduce solicitudes de transacciones a una base de datos remota. Suponga que la entidad de transporte remota está programada para pasar primero las TPDUs a la siguiente capa superior y luego emitir las confirmaciones de recepción. Aun en este caso, la recepción de una confirmación de

	Estrategia usada por el *host* receptor					
	Primero ACK, luego escritura			Primero escritura, luego ACK		
Estrategia usada por el *host* emisor	AC(W)	AWC	C(AW)	C(WA)	WAC	WC(A)
Siempre retransmitir	BIEN	DUP.	BIEN	BIEN	DUP.	DUP.
Nunca retransmitir	PERD.	BIEN	PERD.	PERD.	BIEN	BIEN
Retransmitir en S0	BIEN	DUP.	PERD.	PERD.	DUP.	BIEN
Retransmitir en S1	PERD.	BIEN	BIEN	BIEN	BIEN	DUP.

BIEN = El protocolo funciona correctamente
DUP. = El protocolo genera un mensaje duplicado
PERD. = El protocolo pierde un mensaje

Figura 6-18. Diferentes combinaciones de la estrategia cliente-servidor.

recepción en la máquina del usuario no significa necesariamente que el *host* remoto se quedó encendido suficiente tiempo para actualizar la base de datos. Probablemente es imposible lograr una verdadera confirmación de recepción de extremo a extremo, cuya recepción significa que el trabajo se ha hecho, y cuya falta significa que no. Este punto lo analizan con mayor detalle Saltzer y cols. (1984.)

6.3 UN PROTOCOLO DE TRANSPORTE SENCILLO

Para hacer más concretas las ideas estudiadas hasta ahora, en esta sección estudiaremos detalladamente una capa de transporte de ejemplo. Las primitivas de servicio abstractas que usaremos son las primitivas orientadas a la conexión de la figura 6-2. La elección de estas primitivas hace que el ejemplo sea similar (pero más sencillo) al popular protocolo TCP.

6.3.1 Las primitivas de servicio de ejemplo

Nuestro primer problema es cómo expresar de manera concreta estas primitivas de transporte. CONNECT es fácil: sencillamente tenemos un procedimiento de biblioteca *connect* que puede llamarse con los parámetros adecuados necesarios para establecer una conexión. Los parámetros son los TSAPs locales y remotos. Durante la llamada, el invocador se bloquea (es decir, se suspende) mientras la entidad de transporte intenta establecer la conexión. Si ésta tiene éxito, se desbloquea el invocador y puede comenzar la transmisión de datos.

Cuando un proceso requiere la capacidad de aceptar llamadas entrantes, llama a *listen*, especificando un TSAP en particular en el que quiere escuchar. El proceso entonces se bloquea hasta que algún proceso remoto intenta establecer una conexión con su TSAP.

Observe que este modelo es altamente asimétrico. Un lado es pasivo, pues ejecuta un *listen* y espera hasta que ocurre algo. El otro lado es activo e inicia la conexión. Una pregunta interesante es qué debe hacerse si el lado activo comienza primero. Una estrategia es hacer que falle el intento de conexión si no hay un escuchador en el TSAP remoto. Otra estrategia es hacer que el iniciador se bloquee (posiblemente para siempre) hasta que aparezca un escuchador.

Una alternativa, usada en nuestro ejemplo, es mantener la solicitud de conexión del lado receptor durante cierto intervalo de tiempo. Si un proceso de ese *host* llama a *listen* antes de que expire el temporizador, se establece la conexión; de otro modo se rechaza y se desbloquea al invocador, devolviéndole un error.

Para liberar una conexión, usaremos un procedimiento *disconnect*. Cuando ambos lados se han desconectado, se libera la conexión. En otras palabras, estamos usando un modelo de desconexión simétrico.

La transmisión de datos tiene precisamente el mismo problema que el establecimiento de una conexión: el envío es activo, pero la recepción es pasiva. Usaremos la misma solución para la transmisión de datos que para el establecimiento de la conexión: una llamada activa *send* que transmite datos, y una llamada pasiva *receive* que bloquea hasta que llega una TPDU.

Nuestra definición concreta del servicio consiste, por tanto, en cinco primitivas: CONNECT, LISTEN, DISCONNECT, SEND y RECEIVE. Cada primitiva corresponde exactamente a un procedimiento de biblioteca que ejecuta la primitiva. Los parámetros de las primitivas de servicio y los procedimientos de biblioteca son los siguientes:

```
connum = LISTEN(local)
connum = CONNECT(local, remote)
status = SEND(connum, buffer, bytes)
status = RECEIVE(connum, buffer, bytes)
status = DISCONNECT(connum)
```

La primitiva LISTEN anuncia la disposición del invocador a aceptar solicitudes de conexión dirigidas al TSAP indicado. El usuario de la primitiva se bloquea hasta que se hace un intento de conexión a él. No hay expiración del temporizador.

La primitiva CONNECT toma dos parámetros, un TSAP local (es decir, dirección de transporte), *local*, y un TSAP remoto, *remote,* e intenta establecer una conexión de transporte entre los dos. Si tiene éxito, devuelve en *connum* un número no negativo que sirve para identificar la conexión en llamadas subsiguientes. Si falla, la razón del fracaso se pone en *connum* como número negativo. En nuestro modelo sencillo, cada TSAP puede participar sólo en una conexión de transporte, por lo que una razón posible de falla es que una de las direcciones de transporte ya esté en uso. Algunas otras razones son: *host* remoto caído, dirección local ilegal y dirección remota ilegal.

La primitiva SEND transmite el contenido del búfer como mensaje por la conexión de transporte indicada, posiblemente en varias unidades si es demasiado grande. Los errores posibles, devueltos en *status*, son falta de conexión, dirección de búfer ilegal y cuenta negativa.

La primitiva RECEIVE indica el deseo del invocador de aceptar datos. El tamaño del mensaje de entrada se pone en *bytes*. Si el proceso remoto ha liberado la conexión o la dirección de búfer es ilegal (por ejemplo, fuera del programa del usuario), *status* se establece a un código de error que indica la naturaleza del problema.

La primitiva DISCONNECT termina una conexión de transporte; el parámetro *connum* indica cuál. Los errores posibles son que *connum* pertenezca a otro proceso, o que *connum* no sea un identificador de conexión válido. El código de error, o 0 si todo funcionó bien, se devuelve en *status*.

6.3.2 La entidad de transporte de ejemplo

Antes de ver el código de la entidad de transporte de ejemplo, tenga bien presente que este ejemplo es análogo a los primeros ejemplos presentados en el capítulo 3: su propósito es pedagógico, no una propuesta seria. En aras de la sencillez, se han omitido aquí muchos de los detalles técnicos (como comprobación extensa de errores) necesarios para un sistema de producción.

La capa de transporte hace uso de las primitivas de servicio de red para enviar y recibir las TPDUs. Para este ejemplo, necesitamos escoger las primitivas de servicio de red a usar. Una selección podría haber sido el servicio de datagramas no confiable. No lo hemos seleccionado para mantener sencillo el ejemplo. Con el servicio de datagramas no confiable, el código de transporte habría sido grande y complejo, encargándose principalmente de paquetes perdidos y retrasados. Es más, muchas de estas ideas ya se analizaron con detalle en el capítulo 3.

En cambio, hemos optado por usar un servicio de red confiable orientado a la conexión. De esta manera, podremos enfocarnos a los asuntos de transporte que no ocurren en las capas inferiores. Entre éstos tenemos el establecimiento de conexiones, la liberación de conexiones y la administración de crédito. Un servicio sencillo de transporte construido encima de una red ATM podría verse así.

En general, la entidad de transporte puede ser parte del sistema operativo del *host* o puede ser un paquete de rutinas de biblioteca operando en el espacio de direcciones del usuario. Por sencillez, nuestro ejemplo se ha programado como si fuera un paquete de biblioteca, pero los cambios necesarios para hacerlo parte del sistema operativo son mínimos (principalmente la manera de acceder a los búferes de usuario).

Sin embargo, vale la pena señalar que en este ejemplo la "entidad de transporte" no es en realidad una entidad independiente, sino parte del proceso de usuario. En particular, cuando el usuario ejecuta una primitiva que se bloquea, como LISTEN, toda la entidad de transporte se bloquea también. En tanto que este diseño es adecuado para un *host* con un solo proceso de usuario, en un *host* con varios usuarios sería más natural hacer que la entidad de transporte sea un proceso aparte, diferente de todos los procesos de usuario.

La interfaz con la capa de red se establece mediante los procedimientos *to_net* y *from_net* (no se muestran). Cada uno tiene seis parámetros. Primero viene el identificador de conexión, que se relaciona uno a uno con los circuitos virtuales de la red. A continuación vienen los bits *Q* y *M* que, al estar establecidos en 1, indican "mensaje de control" y "continúan más datos de este mensaje

en el siguiente paquete”, respectivamente. Tras esto tenemos el tipo de paquete, seleccionado entre los seis tipos de paquete listados en la figura 6-19. Por último, tenemos un apuntador a los datos mismos y un entero que indica el número de bytes de datos.

Paquete de red	Significado
CALL REQUEST	Se envía para establecer una conexión
CALL ACCEPTED	Respuesta a CALL REQUEST
CLEAR REQUEST	Se envía para liberar una conexión
CLEAR CONFIRMATION	Respuesta a CLEAR REQUEST
DATA	Sirve para transportar datos
CREDIT	Paquete de control para manejar la ventana

Figura 6-19. Paquetes de capa de red usados en nuestro ejemplo.

En las llamadas a *to_net*, la entidad de transporte llena todos los parámetros para que los lea la capa de red; en las llamadas a *from_net*, la capa de red divide un paquete de entrada antes de pasarlo a la entidad de transporte. Al pasar información como parámetros de procedimiento en lugar de pasar el paquete de salida o entrada mismo, la capa de transporte queda protegida de los detalles del protocolo de capa de red. Si la entidad de transporte intenta enviar un paquete cuando la ventana corrediza del circuito virtual subyacente esté llena, se suspende en *to_net* hasta que haya espacio en la ventana. Este mecanismo es transparente para la entidad de transporte y lo controla la capa de red usando comandos como *enable_transport_layer* y *disable_transport_layer* análogos a los descritos en los protocolos del capítulo 3. La capa de red también realiza la administración de la ventana de la capa de paquetes.

Además de este mecanismo de suspensión transparente, hay procedimientos *sleep* y *wakeup* explícitos (que no se muestran) invocados por la entidad de transporte. El procedimiento *sleep* se invoca cuando la entidad de transporte está bloqueada lógicamente en espera de un evento externo, generalmente la llegada de un paquete. Tras haber llamado a *sleep*, la entidad de transporte (y, por supuesto, el proceso de usuario) deja de ejecutarse.

El código de la entidad de transporte se muestra en la figura 6-20. Cada conexión está siempre en uno de siete estados, a saber:

1. IDLE—No se ha establecido todavía una conexión.
2. WAITING—Se ha ejecutado CONNECT y enviado CALL REQUEST.
3. QUEUED—Ha llegado una CALL REQUEST; aún no hay LISTEN.
4. ESTABLISHED—Se ha establecido la conexión.
5. SENDING—El usuario está esperando permiso de enviar un paquete.
6. RECEIVING—Se ha hecho un RECEIVE.
7. DISCONNECTING—Se ha hecho localmente un DISCONNECT.

Las transiciones entre estados pueden ocurrir al suceder cualquiera de los siguientes eventos: se ejecuta una primitiva, llega un paquete o expira el temporizador.

Los procedimientos mostrados en la figura 6-20 son de dos tipos. Casi todos pueden invocarse directamente desde los programas de usuario. Sin embargo, *packet_arrival* y *clock* son diferentes. Eventos externos los activan espontáneamente: la llegada de un paquete y los pulsos del reloj, respectivamente. De hecho, son rutinas de interrupción. Supondremos que no se invocan mientras está ejecutándose un procedimiento de identidad de transporte; sólo pueden invocarse cuando el proceso de usuario está dormido o ejecutándose fuera de la entidad de transporte. Esta propiedad es crucial para el funcionamiento correcto del código.

La existencia del bit Q (calificador) en el encabezado del paquete nos permite evitar la sobrecarga de un encabezado de protocolo de transporte. Los mensajes de datos ordinarios se envían como paquetes de datos con $Q = 0$. Los mensajes de control de protocolo de transporte, de los cuales sólo hay uno (CREDIT) en nuestro ejemplo, se envían como paquetes de datos con $Q = 1$. Estos mensajes de control son detectados y procesados por la entidad de transporte receptora.

La principal estructura de datos usada por la entidad de transporte es el arreglo *conn,* que tiene un registro por cada conexión potencial. El registro mantiene el estado de la conexión, incluidas las direcciones de transporte en ambos extremos, la cantidad de mensajes enviados y recibidos por la conexión, el estado actual, el apuntador al búfer de usuario, la cantidad de bytes de los mensajes actuales enviados o recibidos hasta el momento, un bit que indica si el usuario remoto ha emitido o no un DISCONNECT, un temporizador y un contador de permisos que sirve para habilitar el envío de mensajes. No todos estos campos se usan en nuestro ejemplo sencillo, pero una entidad de transporte completa los necesitaría todos, y tal vez más. Se supone que cada entrada de *conn* está inicialmente en el estado inactivo (*IDLE*).

Cuando el usuario llama a CONNECT, la capa de red recibe instrucciones para enviar un paquete CALL REQUEST a la máquina remota, y el usuario se pone a dormir. Cuando llega el paquete CALL REQUEST al otro lado, la entidad de transporte se interrumpe para ejecutar *packet_arrival* a fin de comprobar que el usuario local está escuchando en la dirección especificada. De ser así, se devuelve un paquete CALL ACCEPTED y se despierta al usuario remoto; de no ser así, la CALL REQUEST se pone en cola durante *TIMEOUT* pulsos de reloj. Si se hace un LISTEN en este periodo, se establece la conexión; de otro modo, expira el temporizador y se rechaza la solicitud con un paquete CLEAR REQUEST para evitar que se bloquee de manera indefinida.

Aunque hemos eliminado el encabezado del protocolo de transporte, aún necesitamos una manera de llevar el control de cuál paquete pertenece a cuál conexión de transporte, ya que pueden existir varias conexiones al mismo tiempo. El enfoque más sencillo es usar el número de circuito virtual de la capa de red también como número de conexión de transporte. Es más, puede usarse también el número de circuito virtual como índice del arreglo *conn.* Al llegar un paquete por el circuito virtual k de la capa de red, pertenecerá a la conexión de transporte k, cuyo estado está en el registro *conn*[k]. En las conexiones iniciadas en un *host*, la entidad de transporte originadora selecciona el número de conexión. En las llamadas entrantes, la capa de red hace la selección, escogiendo cualquier número de circuito virtual no usado.

Para evitar el hecho de tener que proporcionar y administrar *búferes* en la entidad de transporte, se usa aquí un mecanismo de control de flujo diferente de la ventana corrediza tradicional.

```
#define MAX_CONN 32                    /* número máximo de conexiones simultáneas */
#define MAX_MSG_SIZE 8192              /* mensaje más grande en bytes */
#define MAX_PKT_SIZE 512               /* paquete más grande en bytes */
#define TIMEOUT 20
#define CRED 1
#define OK 0

#define ERR_FULL -1
#define ERR_REJECT -2
#define ERR_CLOSED -3
#define LOW_ERR -3

typedef int transport_address;
typedef enum {CALL_REQ,CALL_ACC,CLEAR_REQ,CLEAR_CONF,DATA_PKT,CREDIT} pkt_type;
typedef enum {IDLE,WAITING,QUEUED,ESTABLISHED,SENDING,RECEIVING,DISCONN} cstate;

/* Variables globales. */
transport_address listen_address;     /* dirección local en la que se está escuchando */
int listen_conn;                       /* identificador de conexión para escuchar */
unsigned char data[MAX_PKT_SIZE];      /* área de trabajo para datos de paquetes */

struct conn {
 transport_address local_address, remote_address;
 cstate state;                         /* estado de esta conexión */
 unsigned char *user_buf_addr;         /* apuntador al búfer de recepción */
 int byte_count;                       /* conteo de envío/recepción */
 int clr_req_received;                 /* establecido al recibir el paquete CLEAR_REQ */
 int timer;                            /* se utiliza para establecer el temporizador
                                          de paquetes CALL_REQ */
 int credits;                          /* cantidad de mensajes que se puede enviar */
}conn[MAX_CONN + 1];                   /* la ranura 0 no se utiliza */

void sleep(void);                      /* prototipos */
void wakeup(void);
void to_net(int cid, int q, int m, pkt_type pt, unsigned char *p, int bytes);
void from_net(int *cid, int *q, int *m, pkt_type *pt, unsigned char *p, int *bytes);

int listen(transport_address t)
{/* El usuario desea escuchar por conexión. Ve si CALL_REQ ya llegó. */
 int i, found = 0;

 for (i = 1; i <= MAX_CONN; i++)       /* busca CALL_REQ en la tabla */
     if (conn[i].state == QUEUED && conn[i].local_address == t) {
          found = i;
          break;
     }

  if (found == 0) {
      /* No hay CALL_REQ en espera. Se duerme hasta que llega o expira el
         temporizador. */
      listen_address = t;  sleep();  i = listen_conn ;
  }
  conn[i].state = ESTABLISHED;         /* se ESTABLECE la conexión */
  conn[i].timer = 0;                   /* no se usa el temporizador */
```

```
  listen_conn = 0;                    /* se supone que 0 es una dirección inválida */
  to_net(i, 0, 0, CALL_ACC, data, 0); /* indica a la red que acepte una conexión */
  return(i);                          /* devuelve el identificador de conexión */
 }

 int connect(transport_address l, transport_address r)
 { /* El usuario desea conectarse a un proceso remoto; envía paquete CALL_REQ. */
  int i;
  struct conn *cptr;

  data[0] = r;   data[1] = l;         /* el paquete CALL_REQ los necesita */
  i = MAX_CONN;                       /* busca hacia atrás en la tabla */
  while (conn[i].state != IDLE && i > 1) i = i - 1;
  if (conn[i].state == IDLE) {
      /* Crea una entrada en la tabla indicando que CALL_REQ se ha enviado. */
      cptr = &conn[i];
      cptr->local_address = l; cptr->remote_address = r;
      cptr->state = WAITING; cptr->clr_req_received = 0;
      cptr->credits = 0; cptr->timer = 0;
      to_net(i, 0, 0, CALL_REQ, data, 2);
      sleep();                        /* espera CALL_ACC o CLEAR_REQ */
      if (cptr->state == ESTABLISHED) return(i);
      if (cptr->clr_req_received) {
           /* El otro lado rechazó la llamada. */
           cptr->state = IDLE;        /* regresa al estado de inactividad */
           to_net(i, 0, 0, CLEAR_CONF, data, 0);
           return(ERR_REJECT);
      }
  } else return(ERR_FULL);            /* rechaza la conexión: no hay espacio en la
                                         tabla */
}

int send(int cid, unsigned char bufptr[], int bytes)
{ /* El usuario desea enviar un mensaje. */
  int i, count, m;
  struct conn *cptr = &conn[cid];

  /* Pasa al estado de transmisión. */
  cptr->state = SENDING;
  cptr->byte_count = 0;               /* # de bytes enviados hasta ahora de este
                                         mensaje */
  if (cptr->clr_req_received == 0 && cptr->credits == 0) sleep();
  if (cptr->clr_req_received == 0) {
       /* Crédito disponible; divide el mensaje en paquetes si es necesario. */
       do {
           if (bytes - cptr->byte_count > MAX_PKT_SIZE) { /* mensaje con múltiples
                                                             paquetes */
                 count = MAX_PKT_SIZE;  m = 1;    /* más paquetes posteriormente */
            } else {                  /* mensaje de un solo paquete */
                 count = bytes - cptr->byte_count;  m = 0; /* último paquete de este
                                                              mensaje*/
            }
            for (i = 0; i < count; i++) data[i] = bufptr[cptr->byte_count + i];
            to_net(cid, 0, m, DATA_PKT, data, count);     /* envía un paquete */
            cptr->byte_count = cptr->byte_count + count;  /* incrementa los bytes
                                                             enviados hasta ahora */
       } while (cptr->byte_count < bytes);  /* repite el ciclo hasta enviar todo el
                                               mensaje */
```

```
      cptr->credits--;              /* cada mensaje utiliza un crédito */
      cptr->state = ESTABLISHED;
      return(OK);
  } else {
      cptr->state = ESTABLISHED;
      return(ERR_CLOSED);           /* transmisión fallida: el igual desea
                                       desconectarse */
  }
}

int receive(int cid, unsigned char bufptr[], int *bytes)
{ /* El usuario está preparado para recibir un mensaje. */
  struct conn *cptr = &conn[cid];

  if (cptr->clr_req_received == 0) {
       /* La conexión aún permanece; trata de recibir. */
       cptr->state = RECEIVING;
       cptr->user_buf_addr = bufptr;
       cptr->byte_count = 0;
       data[0] = CRED;
       data[1] = 1;
       to_net(cid, 1, 0, CREDIT, data, 2); /* envía crédito */
       sleep();                     /* se bloquea en espera de datos */
       *bytes = cptr->byte_count;
  }
  cptr->state = ESTABLISHED;
  return(cptr->clr_req_received ? ERR_CLOSED : OK);
}

int disconnect(int cid)
{ /* El usuario desea liberar una conexión. */
  struct conn *cptr = &conn[cid];

  if (cptr->clr_req_received) {     /* el otro lado inició la terminación */
       cptr->state = IDLE;          /* se libera la conexión */
       to_net(cid, 0, 0, CLEAR_CONF, data, 0);
  } else {                          /* nosotros iniciamos la terminación */
      cptr->state = DISCONN;        /* no liberada hasta que el otro lado esté
                                       de acuerdo */
      to_net(cid, 0, 0, CLEAR_REQ, data, 0);
  }
  return(OK);
}

void packet_arrival(void)
{ /* Ha llegado un paquete, se obtiene y procesa. */
  int cid;                          /* conexión en la cual arribó el paquete */
  int count, i, q, m;
  pkt_type ptype;     /* CALL_REQ, CALL_ACC, CLEAR_REQ, CLEAR_CONF, DATA_PKT, CREDIT */
  unsigned char data[MAX_PKT_SIZE];  /* porción de datos del paquete que llegó */
  struct conn *cptr;

  from_net(&cid, &q, &m, &ptype, data, &count);    /* se obtiene*/
  cptr = &conn[cid];
```

```
  switch (ptype) {
   case CALL_REQ:                    /* el usuario remoto desea establecer conexión */
    cptr->local_address = data[0];  cptr->remote_address = data[1];
    if (cptr->local_address == listen_address) {
          listen_conn = cid;  cptr->state = ESTABLISHED;  wakeup();
    } else {
         cptr->state = QUEUED;  cptr->timer = TIMEOUT;
    }
    cptr->clr_req_received = 0;   cptr->credits = 0;
    break;

   case CALL_ACC:                    /* el usuario remoto ha aceptado nuestra CALL_
                                        REQ */
    cptr->state = ESTABLISHED;
    wakeup();
    break;

   case CLEAR_REQ:                   /* usuario remoto desea desconectarse o rechazar
                                        la llamada */
    cptr->clr_req_received = 1;
    if (cptr->state == DISCONN) cptr->state = IDLE; /* limpia la colisión */
    if (cptr->state == WAITING || cptr->state == RECEIVING || cptr->state == SENDING)
       wakeup();
    break;

   case CLEAR_CONF:                  /* el usuario remoto está de acuerdo en la
                                        desconexión */
    cptr->state = IDLE;
    break;

   case CREDIT:                      /* el usuario remoto espera datos */
    cptr->credits += data[1];
    if (cptr->state == SENDING) wakeup();
    break;

   case DATA_PKT:                    /* el usuario remoto ha enviado datos */
    for (i = 0; i < count; i++) cptr->user_buf_addr[cptr->byte_count + i] = data[i];
    cptr->byte_count += count;
    if (m == 0 ) wakeup();
  }
}
void clock(void)
{ /* El reloj pulsó, verificar si expiró el temporizador de solicitudes de conexión
     en cola. */
  int i;
  struct conn *cptr;
  for (i = 1; i <= MAX_CONN; i++) {
      cptr = &conn[i];
      if (cptr->timer > 0) {         /* el temporizador estaba en funcionamiento */
             cptr->timer--;
             if (cptr->timer == 0) {  /* el temporizador expiró */
                    cptr->state = IDLE;
                    to_net(i, 0, 0, CLEAR_REQ, data, 0);
             }
      }
  }
}
```

Figura 6-20. Entidad de transporte de ejemplo.

Cuando un usuario llama a RECEIVE, se envía un **mensaje de crédito** especial a la entidad de transporte de la máquina emisora y se registra en el arreglo *conn*. Al llamar a SEND, la entidad de transporte revisa si ha llegado un crédito en la conexión especificada. De ser así, el mensaje se envía (en múltiples paquetes de ser necesario) y se disminuye el crédito; en caso contrario, la entidad de transporte se pone a dormir hasta la llegada de un crédito. Este mecanismo garantiza que no se enviará nunca un mensaje a menos que el otro lado ya haya hecho un RECEIVE. Como resultado, siempre que llegue un mensaje habrá un búfer disponible en el cual puede ponerse. El esquema puede generalizarse fácilmente para permitir que los receptores proporcionen múltiples búferes y soliciten múltiples mensajes.

Es importante no olvidar la sencillez de la figura 6-20. Una entidad de transporte para uso real normalmente comprobaría la validez de todos los parámetros proporcionados por el usuario, manejaría la recuperación de caídas de la capa de red, se encargaría de colisiones de llamadas y soportaría un servicio de transporte más general, incluidas funciones como interrupciones, datagramas y versiones no bloqueadoras de las primitivas SEND y RECEIVE.

6.3.3 El ejemplo como máquina de estados finitos

Escribir una entidad de transporte es un trabajo difícil y exigente, especialmente para los protocolos más realistas. A fin de reducir la posibilidad de cometer errores, el estado del protocolo se puede representar como máquina de estados finitos.

Ya hemos visto que nuestro protocolo de ejemplo tiene siete estados por conexión. También es posible aislar 12 eventos que pueden ocurrir para pasar una conexión de un estado a otro. Cinco de estos eventos son las cinco primitivas de servicio. Otros seis son las llegadas de los seis tipos de paquete legales. El último es la expiración del temporizador. En la figura 6-21 se muestran las acciones principales del protocolo en forma de matriz. Las columnas son los estados y las filas son los 12 eventos.

Cada entrada de la matriz (es decir, la máquina de estados finitos) de la figura 6-21 tiene hasta tres campos: un predicado, una acción y un estado nuevo. El predicado indica en qué condiciones se emprende la acción. Por ejemplo, en la entrada de la esquina superior izquierda, si se ejecuta una LISTEN y no hay más espacio de tabla (predicado *P1*) falla la LISTEN y no cambia el estado. Por otra parte, si ya ha llegado un paquete CALL REQUEST para la dirección de transporte en la que se está escuchando (predicado *P2*), se establece de inmediato la conexión. Otra posibilidad es que *P2* sea falso, es decir, que no ha entrado ningún CALL REQUEST, en cuyo caso la conexión permanece en el estado *IDLE*, esperando un paquete CALL REQUEST.

Es importante indicar que la selección de estados a usar en la matriz no es fijada completamente por el protocolo mismo. En este ejemplo, no hay ningún estado *LISTENING*, lo que sería algo razonable después de una LISTEN. No hay estado *LISTENING* porque un estado se asocia a un registro de conexión, y LISTEN no crea un registro de conexión. ¿Por qué no? Porque hemos decidido usar los números de circuito virtual de la capa de red como identificadores de la conexión, y el número de circuito virtual para una LISTEN es escogido por la capa de red al llegar el paquete CALL REQUEST.

		Estado						
		Inactivo	En espera	En cola	Establecido	Trans-mitiendo	Recibiendo	Des-conexión
Primitivas	LISTEN	P1: ~/Inactivo $\overline{P2}$: A1/Estab. P2: A2/Inactivo		~/Estab.				
	CONNECT	$\overline{P1}$: ~/Inactivo P1: A3/En esp.						
	DISCONNECT				$\overline{P4}$: A5/Inactivo P4: A6/Desc.			
	SEND				$\overline{P5}$: A7/Estab. P5: A8/Trans.			
	RECEIVE				A9/Recib.			
Paquetes de entrada	Call_req	$\overline{P3}$: A1/Estab. P3: A4/En cola						
	Call_acc		~/Estab.					
	Clear_req		~/Inactivo		A10/Estab.	A10/Estab.	A10/Estab.	~/Inactivo
	Clear_conf							~/Inactivo
	DataPkt						A12/Estab.	
	Credit				A11/Estab.	A7/Estab.		
Reloj	Timeout			~/Inactivo				

Predicados

P1: Tabla de conexiones llena
P2: Call_req pendiente
P3: LISTEN pendiente
P4: Clear_req pendiente
P5: Crédito disponible

Acciones

A1: Envía Call_acc
A2: Espera Call_req
A3: Envía Call_req
A4: Inicia temporizador
A5: Envía Clear_conf
A6: Envía Clear_req
A7: Envía mensaje
A8: Espera crédito
A9: Envía crédito
A10: Establece indicador Clr_req_received
A11: Registra crédito
A12: Acepta mensaje

Figura 6-21. El protocolo de ejemplo como máquina de estados finitos. Cada entrada tiene un predicado opcional, una acción opcional y el estado nuevo. La tilde indica que no se realizó ninguna acción importante. Una barra sobre un predicado indica la negación del predicado. Las entradas en blanco corresponden a eventos imposibles o inválidos.

Las acciones *A1* a *A12* son las acciones principales, como el envío de paquetes y el inicio de temporizadores. No se listan todas las acciones menores, como la inicialización de los campos de un registro de conexión. Si una acción comprende despertar un proceso dormido, las acciones que

siguen a su despertar también cuentan. Por ejemplo, si entra un paquete CALL REQUEST y un proceso estaba dormido esperándolo, la transmisión del paquete CALL ACCEPT que sigue al despertar cuenta como parte de la acción de CALL REQUEST. Tras la ejecución de cada acción, la conexión puede pasar a un estado nuevo, como se muestra en la figura 6-21.

La ventaja de representar el protocolo como una matriz es triple. Primero, de esta manera es mucho más fácil que el programador revise sistemáticamente cada combinación de estado y evento para ver si se requiere una acción. En las implementaciones de producción se usarían algunas de las combinaciones para manejo de errores. En la figura 6-21 no se hace distinción entre situaciones imposibles e ilegales. Por ejemplo, si una conexión está en el estado *waiting*, el evento DISCONNECT es imposible porque el usuario está bloqueado y no puede ejecutar primitivas. Por otra parte, en el estado *sending* no se esperan paquetes de datos porque no se ha emitido crédito. La llegada de un paquete de datos es un error del protocolo.

La segunda ventaja de la representación en matriz del protocolo tiene que ver con su implementación. Podemos imaginar un arreglo de dos dimensiones en el que el elemento $a[i][j]$ es un apuntador o índice al procedimiento que maneja la ocurrencia del evento i cuando se está en el estado j. Una posible implementación es escribir la entidad de transporte como ciclo corto, esperando un evento en la parte superior del ciclo. Al ocurrir un evento, se localiza la conexión pertinente y se extrae su estado. Sabiendo ahora el evento y el estado, la entidad de transporte simplemente indexa en el arreglo a e invoca el procedimiento adecuado. Este enfoque produce un diseño mucho más regular y sistemático que nuestra entidad de transporte.

La tercera ventaja del enfoque de máquina de estados finitos se aprecia en la descripción del protocolo. En algunos documentos estándares, los protocolos se dan como máquinas de estados finitos similares a la de la figura 6-21. Pasar de este tipo de descripción a una entidad de transporte operante es mucho más fácil si la entidad de transporte también es impulsada por una máquina de estados finitos basada en el estándar.

La desventaja principal del enfoque de máquina de estados finitos es que puede ser más difícil de entender que el ejemplo de programación que usamos inicialmente. Sin embargo, este problema puede resolverse parcialmente dibujando la máquina de estados finitos en forma de grafo, como se hace en la figura 6-22.

6.4 LOS PROTOCOLOS DE TRANSPORTE DE INTERNET: UDP

Internet tiene dos protocolos principales en la capa de transporte, uno orientado y otro no orientado a la conexión. En las siguientes secciones analizaremos los dos. El protocolo no orientado a la conexión es UDP. El protocolo orientado a la conexión es TCP. Empezaremos con UDP porque es básicamente IP con un encabezado corto. También veremos dos aplicaciones de UDP.

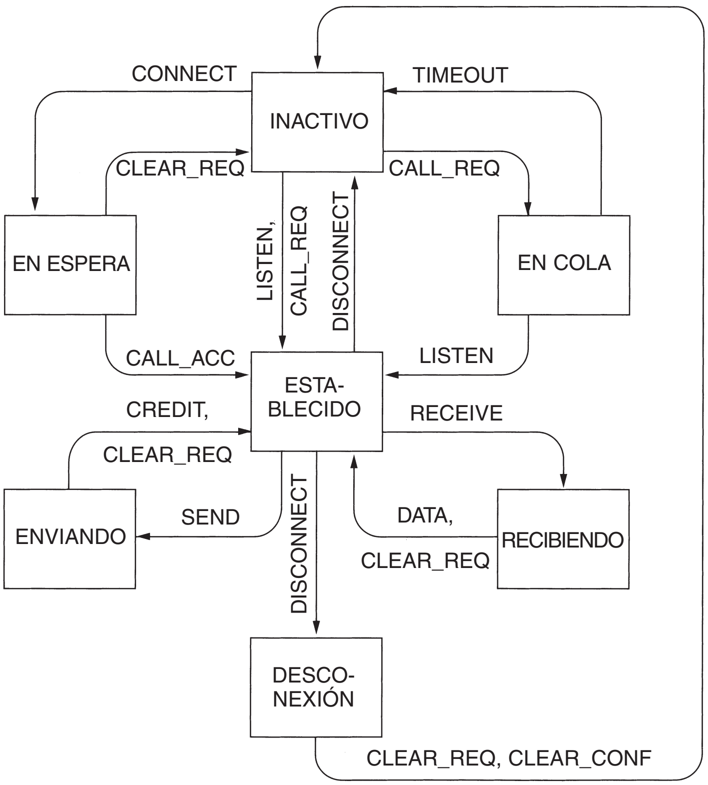


Figura 6-22. El protocolo de ejemplo en forma gráfica. Por sencillez, se han omitido las transiciones que dejan sin cambio el estado de las conexiones.

6.4.1 Introducción a UDP

El conjunto de protocolos de Internet soporta un protocolo de transporte no orientado a la conexión, **UDP** (**Protocolo de Datagramas de Usuario**). Este protocolo proporciona una forma para que las aplicaciones envíen datagramas IP encapsulados sin tener que establecer una conexión. UDP se describe en el RFC 768.

UDP transmite **segmentos** que consisten en un encabezado de 8 bytes seguido por la carga útil. En la figura 6-23 se muestra tal encabezado. Los dos puertos sirven para identificar los puntos terminales dentro de las máquinas de origen y destino. Cuando llega un paquete UDP, su carga útil se entrega al proceso que está enlazado al puerto de destino. Este enlace ocurre cuando se utiliza la primitiva BIND o algo similar, como vimos en la figura 6-6 para TCP (el proceso de enlace es el mismo para UDP). De hecho, el valor principal de contar con UDP en lugar de simplemente utilizar el IP puro es la adición de los puertos de origen y destino. Sin los campos de puerto, la capa de transporte no sabría qué hacer con el paquete. Con ellos, entrega los segmentos de manera correcta.

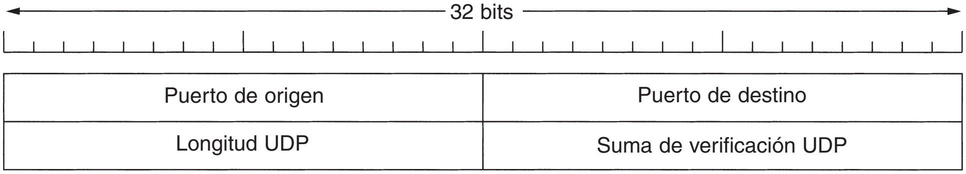


Figura 6-23. El encabezado UDP.

El puerto de origen se necesita principalmente cuando debe enviarse una respuesta al origen. Al copiar el campo *Puerto de origen* del segmento que llega en el campo *Puerto de destino* del segmento que sale, el proceso que envía la respuesta puede especificar cuál proceso de la máquina emisora va a obtenerlo.

El campo *Longitud UDP* incluye el encabezado de 8 bytes y los datos. El campo *Suma de verificación UDP* es opcional y se almacena como 0 si no se calcula (un 0 calculado se almacena como 1s). Su desactivación no tiene sentido a menos que la calidad del servicio de los datos no importe (por ejemplo, en la voz digitalizada).

Probablemente valga la pena mencionar de manera explícita algunas de las cosas que UDP *no* realiza. No realiza control de flujo, control de errores o retransmisión cuando se recibe un segmento erróneo. Todo lo anterior le corresponde a los procesos de usuario. Lo que sí realiza es proporcionar una interfaz al protocolo IP con la característica agregada de desmultiplexar varios procesos utilizando los puertos. Esto es todo lo que hace. Para aplicaciones que necesitan tener control preciso sobre el flujo de paquetes, control de errores o temporización, UDP es lo ideal.

Un área en la que UDP es especialmente útil es en las situaciones cliente-servidor. Con frecuencia, el cliente envía una solicitud corta al servidor y espera una respuesta corta. Si se pierde la solicitud o la respuesta, el cliente simplemente puede terminar y probar nuevamente. El código no sólo es simple, sino que se necesitan muy pocos mensajes (uno en cada dirección) en comparación con un protocolo que requiere una configuración inicial.

Una aplicación que utiliza de esta manera a UDP es DNS (el Sistema de Nombres de Dominio), el cual analizaremos en el capítulo 7. En resumen, un programa que necesita buscar la dirección IP de algún *host*, por ejemplo, *www.cs.berkeley.edu*, puede enviar al servidor DNS un paquete UDP que contenga el nombre de dicho *host*. El servidor responde con un paquete UDP que contiene la dirección IP del *host*. No se necesita configuración por adelantado ni tampoco liberación posterior. Sólo dos mensajes viajan a través de la red.

6.4.2 Llamada a procedimiento remoto

En cierto sentido, enviar un mensaje a un *host* remoto y obtener una respuesta es muy parecido a hacer una llamada a función en un lenguaje de programación. En ambos casos, usted inicia con uno o más parámetros y obtiene un resultado. Esta observación ha llevado a la gente a tratar de que las interacciones de solicitud-respuesta en redes se asignen en forma de llamadas a procedimiento. Esto hace que las aplicaciones de red sean mucho más fáciles de programar y de manejar.

Por ejemplo, imagine un procedimiento llamado *obt_direccion_IP* (*nombre_de_host*) que envía un paquete UDP a un servidor DNS y espera una respuesta, y en caso de que no llegue ninguna con rapidez, termina y lo intenta nuevamente. De esta forma, todos los detalles de la conectividad pueden ocultarse al programador.

El trabajo clave en esta área fue realizado por Birrell y Nelson (1984), quienes sugirieron que se permitiera que los programas invocaran procedimientos localizados en *hosts* remotos. Cuando un proceso en la máquina 1 llama a uno en la máquina 2, el proceso invocador de la primera se suspende y la ejecución del procedimiento se lleva a cabo en la 2. La información se puede transportar desde el invocador al proceso invocado en los parámetros, y se puede regresar en el resultado del procedimiento. El paso de mensajes es transparente para el programador. Esta técnica se conoce como **RPC** (**Llamada a Procedimiento Remoto**) y se ha vuelto la base de muchas aplicaciones de redes. Tradicionalmente, el procedimiento invocador se conoce como cliente y el proceso invocado, como servidor, por lo que aquí utilizaremos esos nombres.

El propósito esencial de RPC es hacer que una llamada a procedimiento remoto sea lo más parecida posible a una local. En la forma más simple, para llamar a un procedimiento remoto, el programa cliente debe enlazarse con un pequeño procedimiento de biblioteca, llamado ***stub* del cliente,** que representa al procedimiento servidor en el espacio de direcciones del cliente. De forma similar, el servidor se enlaza con una llamada a procedimiento denominada ***stub* del servidor.** Estos procedimientos ocultan el hecho de que la llamada a procedimiento desde el cliente al servidor no es local.

En la figura 6-24 se muestran los pasos reales para realizar una RPC. El paso 1 consiste en que el cliente llame al *stub* del cliente. Ésta es una llamada a procedimiento local, y los parámetros se colocan en la pila de la forma tradicional. El paso 2 consiste en que el *stub* del cliente empaca los parámetros en un mensaje y realiza una llamada de sistema para enviar dicho mensaje. El empaquetamiento de los parámetros se conoce como **marshaling.** El paso 3 consiste en que el *kernel* envía el mensaje desde la máquina cliente a la máquina servidor. El paso 4 consiste en que el *kernel* pasa el paquete entrante al *stub* del servidor. Por último, el paso 5 consiste en que el *stub* del servidor llame al procedimiento servidor con parámetros sin *marshaling*. La respuesta sigue la misma ruta en la dirección opuesta.

El elemento clave a notar aquí es que el procedimiento cliente, escrito por el usuario, simplemente realiza una llamada a procedimiento normal (es decir, local) al *stub* del cliente, lo cual tiene el mismo nombre que el procedimiento servidor. Puesto que el procedimiento cliente y el *stub* del cliente están en el mismo espacio de direcciones, los parámetros se pasan de la forma usual. De manera similar, el procedimiento servidor es llamado por un procedimiento en su espacio de direcciones con los parámetros que esperaba. Para el procedimiento servidor, nada es inusual. De esta forma, en lugar de que la E/S se realice en *sockets*, la comunicación de red se realiza simulando una llamada a procedimiento normal.

A pesar de la elegancia conceptual de RPC, hay algunas desventajas ocultas. La más grande es el uso de parámetros de apuntador. Por lo general, pasar un apuntador a un procedimiento no es un problema. El procedimiento invocado puede utilizar el apuntador de la misma manera que el invocador, porque ambos procedimientos habitan el mismo espacio de direcciones virtual. Con

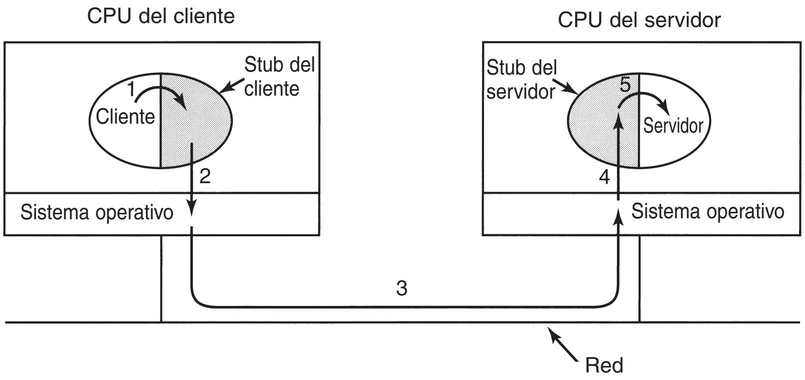


Figura 6-24. Pasos para realizar una llamada a procedimiento remoto. Los stubs están sombreados.

RPC, el paso de apuntadores es imposible porque el cliente y el servidor están en diferentes espacios de direcciones.

En algunos casos, se pueden utilizar trucos para hacer que el paso de apuntadores sea posible. Suponga que el primer parámetro es un apuntador a un entero, *k*. El *stub* del cliente puede clasificar *k* y enviarlo a lo largo del servidor. El *stub* del servidor a continuación crea un apuntador a *k* y lo pasa al procedimiento servidor, justamente como lo espera. Cuando el procedimiento servidor regresa el control al *stub* del servidor, este último regresa *k* al cliente, donde *k* se copia en el anterior, por si el servidor cambió. En efecto, la secuencia de llamada estándar de llamada por referencia se ha reemplazado con copiar-restaurar. Desgraciadamente, este truco no siempre funciona, por ejemplo, si el apuntador apunta hacia un grafo o a otra estructura de datos compleja. Por esta razón, se deben colocar algunas restricciones para los procedimientos llamados de manera remota.

Un segundo problema es que en los lenguajes de tipos flexibles, como C, es perfectamente legal escribir un procedimiento que calcule el producto interno de dos vectores (arreglos), sin especificar la longitud de cada uno. Éstos pueden terminarse mediante un valor especial conocido sólo por los procedimientos invocador e invocado. Bajo estas circunstancias, es esencialmente imposible que el *stub* del cliente aplique *marshaling* a los parámetros debido a que no tiene forma de determinar su longitud.

Un tercer problema es que no siempre es posible deducir los tipos de parámetros, ni siquiera de una especificación formal o el código mismo. Un ejemplo de esto es *printf*, el cual puede tener cualquier número de parámetros (por lo menos uno), y éstos pueden ser una mezcla arbitraria de tipos enteros, cortos, largos, caracteres, cadenas, así como de números de punto flotante de varias longitudes, entre otros. Tratar de llamar a *printf* como un procedimiento remoto sería prácticamente imposible debido a que C es demasiado permisivo. Sin embargo, una regla que especifique el uso de RPC siempre y cuando no se utilice C (o C++) para programar tal vez no sea muy popular.

Un cuarto problema se relaciona con el uso de las variables globales. Por lo general, los procedimientos invocador e invocado pueden comunicarse utilizando variables globales y parámetros. Si el procedimiento invocado se mueve a una máquina remota, el código fallará porque las variables globales ya no se pueden compartir.

Estos problemas no significan que RPC no tiene mucho futuro. De hecho, se utiliza ampliamente, pero se necesitan algunas restricciones para hacerlo funcionar bien en la práctica.

Por supuesto, RPC no necesita utilizar paquetes UDP, pero RPC y UDP son una buena combinación y UDP se utiliza comúnmente con RPC. No obstante, cuando los parámetros o resultados son más grandes que el tamaño máximo del paquete UDP o cuando la operación solicitada no tiene la misma potencia (es decir, no se puede repetir de forma segura, como cuando se incrementa un contador), tal vez sea necesario establecer una conexión TCP y enviar una solicitud a través de ella en lugar de utilizar UDP.

6.4.3 El protocolo de transporte en tiempo real

El RPC cliente-servidor es un área en la que se utiliza ampliamente UDP. Otra son las aplicaciones multimedia en tiempo real. En particular, conforme el radio en Internet, la telefonía en Internet, la música bajo demanda, la videoconferencia, el vídeo bajo demanda y otras aplicaciones multimedia se volvían más comunes, las personas que descubrieron cada una de esas aplicaciones estaba reinventando más o menos el mismo protocolo de transporte de tiempo-real. Gradualmente se volvió más claro que tener un protocolo de transporte genérico para múltiples aplicaciones sería una excelente idea. Y fue por esto que nació el **RTP** (**Protocolo de Transporte en Tiempo Real**). Se describe en el RFC 1889 y ahora se utiliza ampliamente.

La posición del RTP en la pila de protocolos es algo extraña. Se decidió colocarlo en el espacio de usuario y ejecutarlo (por lo general) sobre UDP. Opera como se muestra a continuación. La aplicación multimedia consiste en múltiples flujos de audio, vídeo, texto, entre otros. Éstos se colocan en la biblioteca RTP, la cual está en el espacio de usuario junto con la aplicación. Esta biblioteca multiplexa los flujos y los codifica en paquetes RTP, que después coloca en un *socket*. En el otro extremo del *socket* (en el *kernel* del sistema operativo), los paquetes UDP se generan e incrustan en paquetes IP. Si la computadora está en una Ethernet, los paquetes IP se colocan a continuación en tramas Ethernet para su transmisión. En la figura 6-25(a) se muestra la pila de protocolos para esta situación, y en la 6-25(b) se muestra el anidamiento de paquetes.

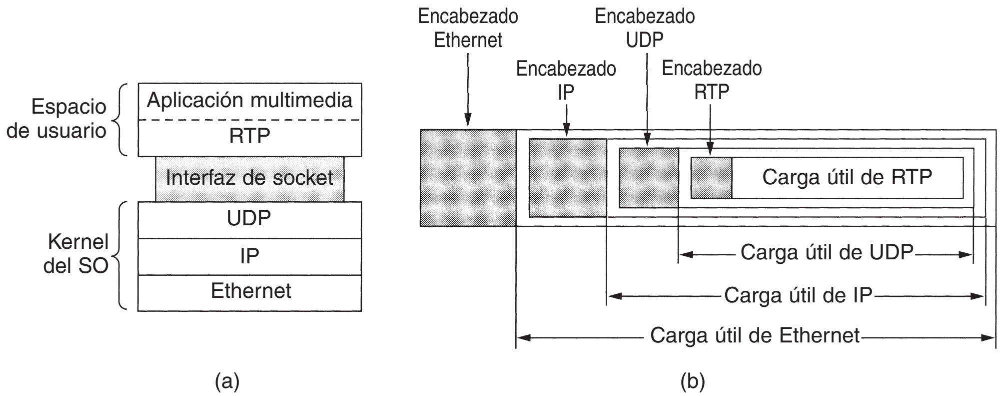


Figura 6-25. (a) La posición del RTP en la pila de protocolos. (b) Anidamiento de paquetes.

Como consecuencia de este diseño, es un poco difícil decir en cuál capa está RTP. Debido a que se ejecuta en el espacio de usuario y a que está enlazado al programa de aplicación, ciertamente luce como un protocolo de aplicación. Por otro lado, es un protocolo genérico, independiente de la aplicación que simplemente proporciona características de transporte, por lo que se parece a un protocolo de transporte. Probablemente la mejor descripción es que es un protocolo de transporte que está implementado en la capa de aplicación.

La función básica de RTP es multiplexar varios flujos de datos en tiempo real en un solo flujo de paquetes UDP. El flujo UDP se puede enviar a un solo destino (unidifusión) o a múltiples destinos (multidifusión). Debido a que RTP sólo utiliza UDP normal, sus paquetes no son tratados de manera especial por los enrutadores, a menos que se habiliten algunas características de calidad de servicio IP normales. En particular, no hay garantías especiales acerca de la entrega, fluctuación, etcétera.

A cada paquete enviado en un flujo RTP se le da un número más grande que a su predecesor. Esta numeración permite al destino determinar si falta algún paquete. Si falta alguno, la mejor acción que el destino puede realizar es aproximar el valor faltante mediante la interpolación. La retransmisión no es una opción práctica debido a que el paquete retransmitido probablemente llegará muy tarde como para ser útil. En consecuencia, RTP no tiene control de flujo, control de errores, confirmaciones de recepción ni ningún mecanismo para solicitar retransmisiones.

Cada carga útil RTP podría contener múltiples muestras, y éstas podrían codificarse de la forma en la que la aplicación desee. Para permitir la interconectividad, RTP define diversos perfiles (por ejemplo, un solo flujo de audio), y para cada perfil se podrían permitir múltiples formatos de codificación. Por ejemplo, un solo flujo de audio podría codificarse como muestras de PCM de 8 bits a 8 kHz, codificación delta, codificación predictiva, codificación GSM, MP3, etcétera. RTP proporciona un campo de encabezado en el que el origen puede especificar la codificación, pero por otro lado no tiene nada que ver en la forma en que se realiza la codificación.

Otra característica que muchas de las aplicaciones en tiempo real necesitan es la marcación del tiempo (*timestamping*). La idea aquí es permitir que el origen asocie una marca de tiempo con la primera muestra de cada paquete. Las marcas de tiempo son relativas al inicio del flujo, por lo que sólo son importantes las diferencias entre dichas marcas de tiempo. Los valores absolutos no tienen significado. Este mecanismo permite que el destino realice una pequeña cantidad de almacenamiento en búfer y reproduzca cada muestra el número exacto de milisegundos después del inicio del flujo, independientemente de cuándo llegó el paquete que contiene la muestra. La marcación del tiempo no sólo reduce los efectos de la fluctuación, sino que también permite que múltiples flujos estén sincronizados entre sí. Por ejemplo, un programa de televisión digital podría tener un flujo de vídeo y dos flujos de audio. Estos flujos de audio podrían ser para difusiones de estéreo o para manejar películas con la banda sonora del idioma original y con una doblada al idioma local, dándole una opción al espectador. Cada flujo proviene de un dispositivo físico diferente, pero si tienen marcas de tiempo de un solo contador, pueden reproducirse de manera síncrona, incluso si los flujos se transmiten de manera irregular.

En la figura 6-26 se ilustra el encabezado RTP. Consiste de tres palabras de 32 bits y de algunas extensiones. La primera palabra contiene el campo *Versión*, que es la 2. Esperemos que esta versión esté muy cerca de la final debido a que sólo quedó pendiente un punto del código

(aunque se puede definir como 3 dando a entender que la versión real estaba en una palabra de extensión).

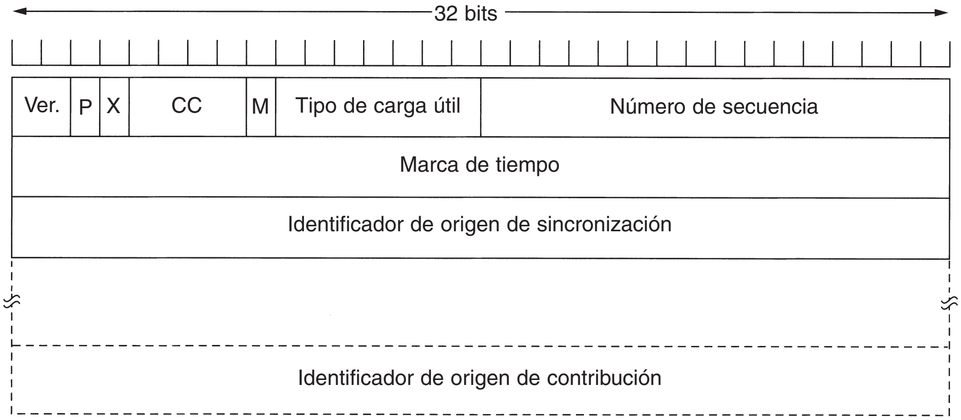


Figura 6-26. El encabezado RTP.

El bit *P* indica que el paquete se ha rellenado a un múltiplo de 4 bytes. El último byte de relleno indica cuántos bytes se agregaron. El bit *X* indica que hay un encabezado de extensión. El formato y el significado de este encabezado no se definen. Lo único que se define es que la primera palabra de la extensión proporciona la longitud. Ésta es una puerta de escape para cualquier requerimiento imprevisto.

El campo *CC* indica cuántos orígenes de contribución están presentes, de 0 a 15 (vea más adelante). El bit *M* es un bit marcador específico de la aplicación. Puede utilizarse para marcar el inicio de un cuadro de vídeo, el inicio de una palabra en un canal de audio, o algo más que la aplicación entienda. El campo *Tipo de carga útil* indica cuál algoritmo de codificación se ha utilizado (por ejemplo, audio de 8 bits sin compresión, MP3, etcétera). Puesto que cada paquete lleva este campo, la codificación puede cambiar durante la transmisión. El *Número de secuencia* es simplemente un contador que se incrementa en cada paquete RTP enviado. Se utiliza para detectar paquetes perdidos.

El origen del flujo produce la marca de tiempo para indicar cuándo se creó la primera muestra en el paquete. Este valor puede ayudar a reducir la fluctuación en el receptor al desacoplar la reproducción del tiempo de llegada del paquete. El *Identificador de origen de sincronización* indica a cuál flujo pertenece el paquete. Es el método utilizado para multiplexar y desmultiplexar varios flujos de datos en un solo flujo de paquetes UDP. Por último, los *Identificadores de origen de contribución*, en caso de que haya, se utilizan cuando los mezcladores están presentes en el estudio. En ese caso, el mezclador es el origen de sincronización, y los flujos que se mezclan se listan aquí.

RTP tiene un hermano pequeño llamado **RTCP** (**Protocolo de Control de Transporte en Tiempo Real**). Maneja la retroalimentación, sincronización y la interfaz de usuario, pero no transporta ningún tipo de datos. La primera función se puede utilizar para proporcionar a los orígenes

retroalimentación en caso de retardo, fluctuación, ancho de banda, congestión y otras propiedades de red. El proceso de codificación puede utilizar esta información para incrementar la tasa de datos (y para proporcionar mejor calidad) cuando la red está funcionando bien y para disminuir la tasa de datos cuando hay problemas en la red. Al proporcionar retroalimentación continua, los algoritmos de codificación se pueden adaptar continuamente para proporcionar la mejor calidad posible bajo las circunstancias actuales. Por ejemplo, si el ancho de banda crece o decrece durante la transmisión, la codificación puede cambiar de MP3 a PCM de 8 bits a codificación delta conforme se requiera. El campo *Tipo de carga útil* se utiliza para indicar al destino cuál algoritmo de codificación se emplea en el paquete actual, lo que hace posible modificarlo a solicitud.

RTCP también maneja sincronización entre flujos. El problema es que flujos diferentes pueden utilizar relojes diferentes, con distintas granularidades y distintas tasas de derivación. RTCP puede utilizarse para mantenerlos sincronizados.

Por último, RTCP proporciona una forma para nombrar los diversos orígenes (por ejemplo, en texto ASCII). Esta información puede desplegarse en la pantalla del receptor para indicar quién está hablando en ese momento.

Podrá encontrar más información en (Perkins, 2002).

6.5 LOS PROTOCOLOS DE TRANSPORTE DE INTERNET: TCP

UDP es un protocolo simple y tiene algunos usos específicos, como interacciones cliente-servidor y multimedia, pero para la mayoría de las aplicaciones de Internet se necesita una entrega en secuencia confiable. UDP no puede proporcionar esto, por lo que se necesita otro protocolo. Se llama TCP y es el más utilizado en Internet. A continuación lo estudiaremos con detalle.

6.5.1 Introducción a TCP

TCP (**Protocolo de Control de Transmisión**) se diseño específicamente para proporcionar un flujo de bytes confiable de extremo a extremo a través de una interred no confiable. Una interred difiere de una sola red debido a que diversas partes podrían tener diferentes topologías, anchos de banda, retardos, tamaños de paquete y otros parámetros. TCP tiene un diseño que se adapta de manera dinámica a las propiedades de la interred y que se sobrepone a muchos tipos de fallas.

TCP se definió formalmente en el RFC 793. Conforme el tiempo pasó, se detectaron varios errores e inconsistencias, y los requerimientos de algunas áreas cambiaron. En el RFC 1122 se detallan estas especificaciones y algunos arreglos de errores. En el RFC 1323 se dan algunas extensiones.

Cada máquina que soporta TCP tiene una entidad de transporte TCP, ya sea un procedimiento de biblioteca, un proceso de usuario o parte del *kernel*. En todos los casos, maneja flujos TCP e interactúa con la capa IP. Una entidad TCP acepta flujos de datos de usuario de procesos locales, los divide en fragmentos que no excedan los 64 KB (en la práctica, por lo general, 1460 bytes de datos que se ajustan en una sola trama Ethernet con los encabezados IP y TCP), y envía cada fragmento como un datagrama IP independiente. Cuando los datagramas que contienen datos TCP

llegan a una máquina, se pasan a la entidad TCP, la cual reconstruye los flujos de bytes originales. Por simplicidad, algunas veces utilizaremos "TCP" para referirnos a la entidad de transporte TCP (una pieza de software) o al protocolo TCP (un conjunto de reglas). El contexto dejará claro a que nos referimos. Por ejemplo, en la frase "El usuario proporciona los datos a TCP", es claro que nos referimos a la entidad de transporte TCP.

La capa IP no proporciona ninguna garantía de que los datagramas se entregarán de manera apropiada, por lo que corresponde a TCP terminar los temporizadores y retransmitir los datagramas conforme sea necesario. Los datagramas que llegan podrían hacerlo en el orden incorrecto; también corresponde a TCP reensamblarlos en mensajes en la secuencia apropiada. En resumen, TCP debe proporcionar la confiabilidad que la mayoría de los usuarios desean y que IP no proporciona.

6.5.2 El modelo del servicio TCP

El servicio TCP se obtiene al hacer que tanto el servidor como el cliente creen puntos terminales, llamados *sockets*, como se mencionó en la sección 6.1.3. Cada *socket* tiene un número (dirección), que consiste en la dirección IP del *host,* y un número de 16 bits, que es local a ese *host*, llamado **puerto.** Un puerto es el nombre TCP para un TSAP. Para obtener el servicio TCP, se debe establecer de manera explícita una conexión entre un *socket* en la máquina emisora y uno en la máquina receptora. Las llamadas de *socket* se listan en la figura 6-5.

Un *socket* puede utilizarse para múltiples conexiones al mismo tiempo. En otras palabras, dos o más conexiones pueden terminar en el mismo *socket*. Las conexiones se identifican mediante los identificadores de *socket* de los dos extremos, es decir (*socket1*, *socket2*). No se utiliza ningún otro número de circuitos virtuales ni identificador.

Los números de puerto menores que 1024 se llaman **puertos bien conocidos** y se reservan para servicios estándar. Por ejemplo, cualquier proceso que desee establecer una conexión a un *host* para transferir un archivo utilizando FTP puede conectarse con el puerto 21 del *host* de destino para conectar su demonio (*daemon*) FTP. La lista de puertos bien conocidos se proporciona en *www.iana.org*. Se han asignado aproximadamente 300. En la figura 6-27 se listan algunos de los más conocidos.

Ciertamente podría ser posible que el demonio FTP se conecte a sí mismo al puerto 21 en tiempo de arranque, que el demonio telnet se conecte a sí mismo al puerto 23 en tiempo de arranque, y así sucesivamente. Sin embargo, hacer lo anterior podría llenar la memoria con demonios que están inactivos la mayor parte del tiempo. En su lugar, lo que se hace generalmente es que un solo demonio, llamado **inetd** (**demonio de Internet**) en UNIX, se conecte a sí mismo a múltiples puertos y esperar la primera conexión entrante. Cuando eso ocurre, inetd bifurca un nuevo proceso y ejecuta el demonio apropiado en él, dejando que ese demonio maneje la solicitud. De esta forma, los demonios distintos a inetd sólo están activos cuando hay trabajo para ellos. Inetd consulta un archivo de configuración para saber cuál puerto utilizar. En consecuencia, el administrador del sistema puede configurar el sistema para tener demonios permanentes en los puertos más ocupados (por ejemplo, el puerto 80) e inetd en los demás.

Puerto	Protocolo	Uso
21	FTP	Transferencia de archivos
23	Telnet	Inicio remoto de sesión
25	SMTP	Correo electrónico
69	TFTP	Protocolo de transferencia de archivos trivial
79	Finger	Búsqueda de información sobre un usuario
80	HTTP	World Wide Web
110	POP-3	Acceso remoto al correo electrónico
119	NNTP	Noticias USENET

Figura 6-27. Algunos puertos asignados.

Todas las conexiones TCP son de dúplex total y de punto a punto. Dúplex total significa que el tráfico puede ir en ambas direcciones al mismo tiempo. Punto a punto significa que cada conexión tiene exactamente dos puntos finales. TCP no soporta la multidifusión ni la difusión.

Una conexión TCP es un flujo de bytes, no uno de mensajes. Los límites de los mensajes no se preservan de extremo a extremo. Por ejemplo, si el proceso emisor realiza cuatro escrituras de 512 bytes en un flujo TCP, tal vez estos datos se entreguen al proceso receptor como cuatro fragmentos de 512 bytes, dos fragmentos de 1024 bytes, uno de 2048 bytes (vea la figura 6-28), o de alguna otra forma. No hay manera de que el receptor detecte la(s) unidad(es) en la(s) que se escribieron los datos.

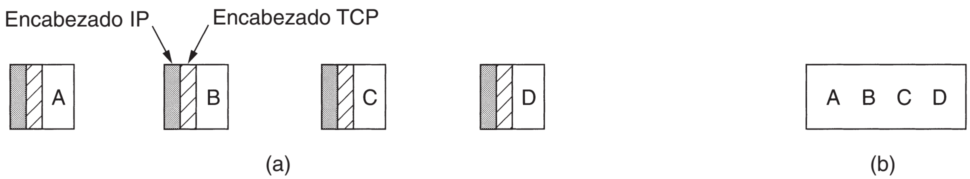


Figura 6-28. (a) Cuatro segmentos de 512 bytes enviados como datagramas IP independientes. (b) Los 2048 bytes de datos que se entregan a la aplicación en una sola llamada a READ.

Los archivos de UNIX también tienen esta propiedad. El lector de un archivo no puede indicar si éste escribió un bloque a la vez, un byte a la vez o todos al mismo tiempo. Al igual que con un archivo de UNIX, el software TCP no tiene idea de lo que significan los bytes y no le interesa averiguarlo. Un byte es sólo un byte.

Cuando una aplicación pasa datos a TCP, éste decide si los envía inmediatamente o los almacena en el búfer (a fin de recolectar una gran cantidad y, así, enviarlos al mismo tiempo). Sin embargo, algunas veces, la aplicación realmente necesita que los datos se envíen de inmediato. Por ejemplo, suponga que un usuario inicia una sesión en una máquina remota. Una vez que se termina una línea de comandos y que se introduce un retorno de carro, es esencial que la línea se envíe

a la máquina remota inmediatamente y que no se almacene en el búfer hasta que llegue la siguiente línea. Para obtener los datos, las aplicaciones pueden utilizar el indicador PUSH, que es una señal para TCP de que no debe retrasar la transmisión.

Algunas de las primeras aplicaciones utilizaban el indicador PUSH como un tipo de marcador para delinear los límites de los mensajes. Si bien este truco funcionaba algunas veces, otras fallaba debido a que no todas las implementaciones de TCP pasan el indicador PUSH a la aplicación del receptor. Además, si llegan indicadores PUSH antes de que el primero se haya transmitido (por ejemplo, debido a que la línea de salida está ocupada), TCP es libre de recolectar todos los datos con indicadores PUSH en un solo datagrama IP, sin ninguna separación entre las diversas piezas.

Una última característica del servicio TCP que vale la pena mencionar son los **datos urgentes.** Cuando un usuario interactivo oprime las teclas Supr o Ctrl+C para interrumpir una operación remota que ha iniciado, la aplicación emisora coloca información de control en el flujo de datos y se la da a TCP junto con el indicador URGENT. Este evento ocasiona que TCP interrumpa el encolamiento de datos y transmita inmediatamente todo lo que tenga para esa conexión.

Cuando el destino recibe los datos urgentes, se interrumpe la aplicación receptora (por ejemplo, se le da una señal en términos de UNIX), a fin de que pueda detener lo que esté haciendo y que lea el flujo de datos para encontrar los datos urgentes. El final de los datos urgentes se marca para que la aplicación sepa cuándo terminan. El inicio de éstos no se marca; la aplicación tiene que averiguarlo. Este esquema proporciona básicamente un mecanismo de señalización simple y deja todo lo demás a la aplicación.

6.5.3 El protocolo TCP

En esta sección daremos un repaso general del protocolo TCP; en la siguiente veremos el encabezado del protocolo, campo por campo.

Una característica clave de TCP, y una que domina el diseño del protocolo, es que cada byte de una conexión TCP tiene su propio número de secuencia de 32 bits. Cuando Internet comenzó, las líneas entre los enrutadores eran principalmente líneas alquiladas de 56 kbps, por lo que un *host* que mandaba datos a toda velocidad tardaba una semana en recorrer los números de secuencia. A las velocidades de las redes modernas, los números de secuencia pueden consumirse con una rapidez alarmante, como veremos más adelante. Los números de secuencia separados de 32 bits se utilizan para confirmaciones de recepción y para el mecanismo de ventana, como se analiza a continuación.

La entidad TCP emisora y la receptora intercambian datos en forma de segmentos. Un **segmento** consiste en un encabezado TCP fijo de 20 bytes (más una parte opcional) seguido de cero o más bytes de datos. El *software* de TCP decide el tamaño de los segmentos; puede acumular datos de varias escrituras para formar un segmento, o dividir los datos de una escritura en varios segmentos. Hay dos límites que restringen el tamaño de segmento. Primero, cada segmento, incluido el encabezado TCP, debe caber en la carga útil de 65,515 bytes del IP. Segundo, cada red tiene una **unidad máxima de transferencia** (**MTU**) y cada segmento debe caber en la MTU. En la práctica, la MTU es, generalmente, de 1500 bytes (el tamaño de la carga útil en Ethernet) y, por tanto, define el límite superior del tamaño de segmento.

El protocolo básico usado por las entidades TCP es el protocolo de ventana corrediza. Cuando un transmisor envía un segmento, también inicia un temporizador. Cuando llega el segmento al destino, la entidad TCP receptora devuelve un segmento (con datos, si existen, de otro modo sin ellos) que contiene un número de confirmación de recepción igual al siguiente número de secuencia que espera recibir. Si el temporizador del emisor expira antes de la recepción de la confirmación, el emisor envía de nuevo el segmento.

Aunque este protocolo suena sencillo, tiene muchos vericuetos que explicaremos a continuación. Por ejemplo, pueden llegar segmentos fuera de orden, por lo que los bytes 3072−4095 podrían llegar pero no enviarse confirmación de recepción porque los bytes 2048−3071 no han aparecido aún. También pueden retardarse segmentos en tránsito durante tanto tiempo que el temporizador del emisor expira y los segmentos se retransmiten. Las retransmisiones podrían incluir rangos de bytes diferentes a los de la transmisión original, lo cual requiere una administración cuidadosa para llevar el control de los bytes que se han recibido correctamente en un momento determinado. Sin embargo, esto es factible ya que cada byte del flujo tiene su propio desplazamiento único.

El TCP debe estar preparado para manejar y resolver estos problemas de una manera eficiente. Se ha invertido una cantidad considerable de esfuerzo en la optimización del desempeño de los flujos TCP, incluso ante problemas de red. A continuación se estudiarán varios de los algoritmos usados por muchas implementaciones de TCP.

6.5.4 El encabezado del segmento TCP

En la figura 6-29 se muestra la distribución de un segmento TCP. Cada segmento comienza con un encabezado de formato fijo de 20 bytes. El encabezado fijo puede ir seguido de opciones de encabezado. Tras las opciones, si las hay, pueden continuar hasta 65,535 − 20 − 20 = 65,495 bytes de datos, donde los primeros 20 se refieren al encabezado IP y los segundos al encabezado TCP. Los segmentos sin datos son legales y se usan por lo común para confirmaciones de recepción y mensajes de control.

Realicemos la disección del encabezado TCP campo por campo. Los campos de *Puerto de origen* y *Puerto de destino* identifican los puntos terminales locales de la conexión. Los puertos bien conocidos se especifican en *www.iana.org* pero cada *host* puede asignar los demás según sea necesario. La dirección de un puerto más la dirección IP de su *host* forman un punto terminal único de 48 bits. Los puntos terminales de origen y de destino en conjunto identifican la conexión.

Los campos de *Número de secuencia* y *Número de confirmación de recepción* desempeñan sus funciones normales. Nótese que el segundo especifica el siguiente byte esperado, no el último byte correctamente recibido. Ambos tienen 32 bits de longitud porque cada byte de datos está numerado en un flujo TCP.

La *Longitud del encabezado TCP* indica la cantidad de palabras de 32 bits contenidas en el encabezado TCP. Esta información es necesaria porque el campo de *Opciones* es de longitud variable, por lo que el encabezado también. Técnicamente, este campo en realidad indica el comienzo

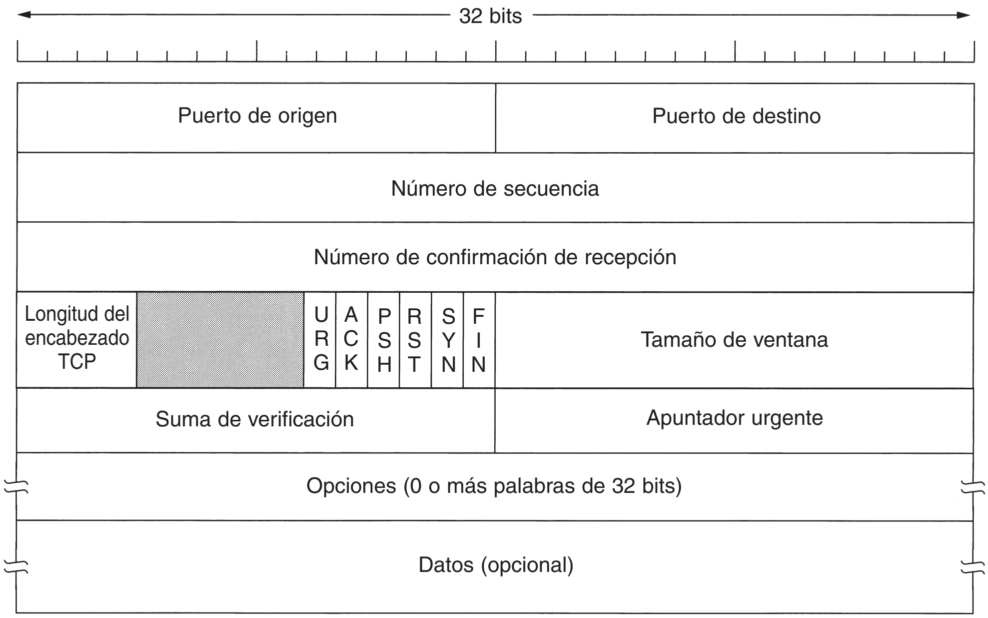


Figura 6-29. Encabezado TCP.

de los datos en el segmento, medido en palabras de 32 bits, pero ese número es simplemente la longitud del encabezado en palabras, por lo que el efecto es el mismo.

A continuación viene un campo de 6 bits que no se usa. El que este campo haya sobrevivido intacto durante más de una década es testimonio de lo bien pensado que está el TCP. Protocolos inferiores lo habrían necesitado para corregir errores del diseño original.

Ahora vienen seis indicadores de 1 bit. *URG* se establece en 1 se está en uso el *apuntador urgente.* El *apuntador urgente* sirve para indicar un desplazamiento en bytes a partir del número actual de secuencia en el que se encuentran datos urgentes. Este recurso sustituye los mensajes de interrupción. Como se mencionó antes, este recurso es un mecanismo rudimentario para permitir que el emisor envíe una señal al receptor sin implicar al TCP en la razón de la interrupción.

El bit *ACK* se establece en 1 para indicar que el *Número de confirmación de recepción* es válido. Si el *ACK* es 0, el segmento no contiene una confirmación de recepción, por lo que se ignora el campo de *Número de confirmación de recepción*.

El bit *PSH* indica datos que se deben transmitir de inmediato. Por este medio se solicita atentamente al receptor que entregue los datos a la aplicación a su llegada y no los almacene en búfer hasta la recepción de un búfer completo (lo que podría hacer en otras circunstancias por razones de eficiencia).

El bit *RST* se usa para restablecer una conexión que se ha confundido debido a una caída de *host* u otra razón; también sirve para rechazar un segmento no válido o un intento de abrir una conexión. Por lo general, si usted recibe un segmento con el bit *RST* encendido, tiene un problema entre manos.

El bit *SYN* se usa para establecer conexiones. La solicitud de conexión tiene *SYN* = 1 y *ACK* = 0 para indicar que el campo de confirmación de recepción incorporado no está en uso. La respuesta de conexión sí lleva una confirmación de recepción, por lo que tiene *SYN* = 1 y *ACK* = 1. En esencia, el bit *SYN* se usa para denotar CONNECTION REQUEST y CONNECTION ACCEPTED, y el bit *ACK* sirve para distinguir entre ambas posibilidades.

El bit *FIN* se usa para liberar una conexión; especifica que el emisor no tiene más datos que transmitir. Sin embargo, tras cerrar una conexión, un proceso puede continuar recibiendo datos indefinidamente. Ambos segmentos, *SYN* y *FIN*, tienen números de secuencia y, por tanto, tienen la garantía de procesarse en el orden correcto.

El control de flujo en el TCP se maneja usando una ventana corrediza de tamaño variable. El campo *Tamaño de ventana* indica la cantidad de bytes que pueden enviarse comenzando por el byte cuya recepción se ha confirmado. Es válido un campo de *Tamaño de ventana* de 0, e indica que se han recibido los bytes hasta *Número de confirmación de recepción* − 1, inclusive, pero que el receptor actualmente necesita un descanso y quisiera no recibir más datos por el momento, gracias. El permiso para enviar puede otorgarse después enviando un segmento con el mismo *Número de confirmación de recepción* y un campo *Tamaño de ventana* distinto de cero.

En los protocolos del capítulo 3, las confirmaciones de recepción de las tramas recibidas y los permisos para enviar nuevas tramas estaban enlazados. Ésta fue una consecuencia de un tamaño de ventana fijo para cada protocolo. En TCP, las confirmaciones de recepción y los permisos para enviar datos adicionales son completamente independientes. En efecto, un receptor puede decir: "He recibido bytes hasta *k*, pero por ahora no deseo más". Esta independencia (de hecho, una ventana de tamaño variable) da flexibilidad adicional. Más adelante lo estudiaremos con más detalle.

También se proporciona una *Suma de verificación* para agregar confiabilidad. Es una suma de verificación del encabezado, los datos y el pseudoencabezado conceptual mostrado en la figura 6-30. Al realizar este cálculo, se establece el campo de *Suma de verificación* del TCP en cero, y se rellena el campo de datos con un byte cero adicional si la longitud es un número impar. El algoritmo de suma de verificación simplemente suma todas las palabras de 16 bits en complemento a 1 y luego obtiene el complemento a 1 de la suma. Como consecuencia, al realizar el cálculo el receptor con el segmento completo, incluido el campo de *Suma de verificación,* el resultado debe ser 0.

El pseudoencabezado contiene las direcciones IP de 32 bits de las máquinas de origen y de destino, el número de protocolo de TCP (6), y la cuenta de bytes del segmento TCP (incluido el encabezado). La inclusión del pseudoencabezado en el cálculo de la suma de verificación TCP ayuda a detectar paquetes mal entregados, pero hacerlo viola la jerarquía de protocolos puesto que las direcciones de IP que contiene pertenecen a la capa IP, no a la capa TCP. UDP utiliza el mismo pseudoencabezado para su suma de verificación.

El campo *Opciones* ofrece una forma de agregar características extra no cubiertas por el encabezado normal. La opción más importante es la que permite que cada *host* especifique la carga útil TCP máxima que está dispuesto a aceptar. El uso de segmentos grandes es más eficiente que el de segmentos pequeños, puesto que el encabezado de 20 bytes puede entonces amortizarse entre más datos, pero los *hosts* pequeños tal vez no puedan manejar segmentos muy grandes. Durante el

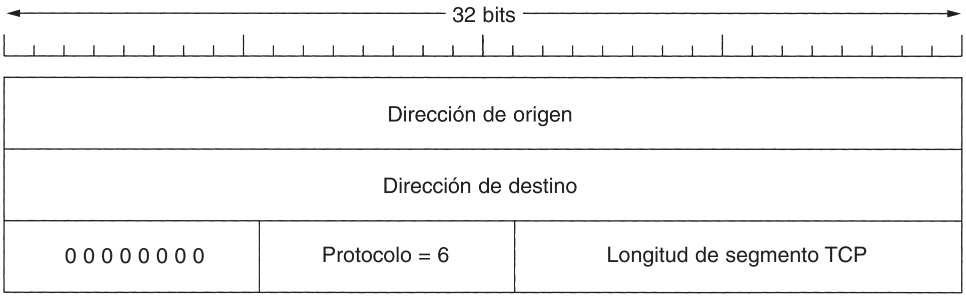


Figura 6-30. Pseudoencabezado incluido en la suma de verificación del TCP.

establecimiento de la conexión, cada lado puede anunciar su máximo y ver el de su compañero. Si un *host* no usa esta opción, tiene una carga útil predeterminada de 536 bytes. Se requiere que todos los *hosts* de Internet acepten segmentos TCP de 536 + 20 = 556 bytes. No es necesario que el tamaño máximo de segmento en ambas direcciones sea el mismo.

En las líneas con alto ancho de banda, alto retardo o ambas cosas, la ventana de 64 KB con frecuencia es un problema. En una línea T3 (44.736 Mbps) se requieren sólo 12 mseg para enviar una ventana completa de 64 KB. Si el retardo de propagación de ida y vuelta es de 50 mseg (típico de una fibra transcontinental), el emisor estará inactivo 3/4 del tiempo en espera de confirmaciones de recepción. En una conexión satelital la situación es peor aún. Un tamaño de ventana más grande permitirá al emisor continuar enviando datos, pero como el campo de *Tamaño de ventana* es de 16 bits, es imposible expresar tal tamaño. En el RFC 1323 se propuso una opción de *escala de ventana* que permite al emisor y al receptor negociar un factor de escala de ventana. Este número da la posibilidad de que ambos lados desplacen el campo de *Tamaño de ventana* hasta 14 bits a la izquierda, permitiendo por tanto ventanas de hasta 2^{30} bytes. La mayoría de las implementaciones actuales de TCP manejan esta opción.

Otra opción propuesta en el RFC 1106 y ahora de uso difundido es el empleo de la repetición selectiva en lugar del protocolo de retroceso *n*. Si el receptor recibe un segmento malo y luego una gran cantidad de segmentos buenos, el temporizador del protocolo TCP normal expirará en algún momento y se retransmitirán todos los segmentos sin confirmación de recepción, incluidos los que se recibieron correctamente. El RFC 1106 introdujo los NAKs, para permitir que el receptor solicite un segmento (o segmentos) específico. Tras recibirlo, puede enviar una confirmación de recepción de todos los datos que tiene en búfer, reduciendo de esta manera la cantidad de datos retransmitidos.

6.5.5 Establecimiento de una conexión TCP

En el TCP las conexiones se establecen usando el acuerdo de tres vías estudiado en la sección 6.2.2. Para establecer una conexión, un lado, digamos el servidor, espera pasivamente una conexión entrante ejecutando las primitivas LISTEN y ACCEPT y especificando cierto origen o bien nadie en particular.

El otro lado, digamos el cliente, ejecuta una primitiva CONNECT especificando la dirección y el puerto IP con el que se desea conectar, el tamaño máximo de segmento TCP que está dispuesto a aceptar y opcionalmente algunos datos de usuario (por ejemplo, una contraseña). La primitiva CONNECT envía un segmento TCP con el bit *SYN* encendido y el bit *ACK* apagado, y espera una respuesta.

Al llegar el segmento al destino, la entidad TCP ahí revisa si hay un proceso que haya ejecutado un LISTEN en el puerto indicado en el campo de *Puerto de destino*. Si no lo hay, envía una respuesta con el bit *RST* encendido para rechazar la conexión.

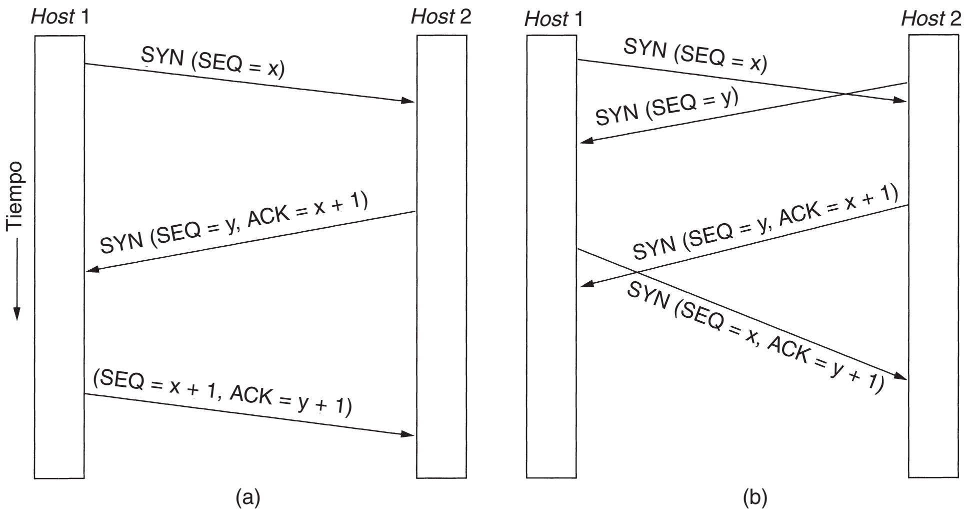


Figura 6-31. (a) Establecimiento de una conexión TCP en el caso normal. (b) Colisión de llamadas.

Si algún proceso está escuchando en el puerto, ese proceso recibe el segmento TCP entrante y puede entonces aceptar o rechazar la conexión; si la acepta, se devuelve un segmento de confirmación de recepción. La secuencia de segmentos TCP enviados en el caso normal se muestra en la figura 6-31(a). Nótese que un segmento *SYN* consume 1 byte de espacio de secuencia, por lo que puede reconocerse sin ambigüedades.

En el caso en que dos *hosts* intentan simultáneamente establecer una conexión entre los mismos dos *sockets*, la secuencia de eventos es la que se ilustra en la figura 6-31(b). El resultado de estos eventos es que sólo se establece una conexión, no dos, pues las conexiones se identifican por sus puntos terminales. Si el primer establecimiento resulta en una conexión identificada por (*x, y*), al igual que en el segundo, sólo se hace una entrada de tabla, es decir, de (*x, y*).

El número de secuencia inicial de una conexión no es 0 por las razones que señalamos antes. Se usa un esquema basado en reloj, con un pulso de reloj cada 4 μseg. Por seguridad adicional, al caerse un *host,* no podrá reiniciarse durante el tiempo máximo de paquete (120 seg) para asegurar que no haya paquetes de conexiones previas vagando por Internet.

6.5.6 Liberación de una conexión TCP

Aunque las conexiones TCP son dúplex total, para entender la manera en que se liberan las conexiones es mejor visualizarlas como un par de conexiones símplex. Cada conexión símplex se libera independientemente de su igual. Para liberar una conexión, cualquiera de las partes puede enviar un segmento TCP con el bit *FIN* establecido, lo que significa que no tiene más datos por transmitir. Al confirmarse la recepción del *FIN*, ese sentido se apaga. Sin embargo, puede continuar un flujo de datos indefinido en el otro sentido. Cuando ambos sentidos se han apagado, se libera la conexión. Normalmente se requieren cuatro segmentos TCP para liberar una conexión, un *FIN* y un *ACK* para cada sentido. Sin embargo, es posible que el primer *ACK* y el segundo *FIN* estén contenidos en el mismo segmento, reduciendo la cuenta total a tres.

Al igual que con las llamadas telefónicas en las que ambas partes dicen adiós y cuelgan el teléfono simultáneamente, ambos extremos de una conexión TCP pueden enviar segmentos *FIN* al mismo tiempo. La recepción de ambos se confirma de la manera normal, y se apaga la conexión. De hecho, en esencia no hay diferencia entre la liberación secuencial o simultánea por parte de los *hosts.*

Para evitar el problema de los dos ejércitos, se usan temporizadores. Si no llega una respuesta a un *FIN* en un máximo de dos tiempos de vida de paquete, el emisor del *FIN* libera la conexión. Tarde o temprano el otro lado notará que, al parecer, ya nadie lo está escuchando, y también expirará su temporizador. Aunque esta solución no es perfecta, dado el hecho de que teóricamente es imposible una solución perfecta tendremos que conformarnos con ella. En la práctica, pocas veces ocurren problemas.

6.5.7 Modelado de administración de conexiones TCP

Los pasos requeridos para establecer y liberar conexiones pueden representarse en una máquina de estados finitos con los 11 estados listados en la figura 6-32. En cada estado son legales ciertos eventos. Al ocurrir un evento legal, debe emprenderse alguna acción. Si ocurren otros eventos, se informa un error.

Cada conexión comienza en el estado *CLOSED* (cerrado) y deja ese estado cuando hace una apertura pasiva (LISTEN), o una apertura activa (CONNECT). Si el otro lado realiza la acción opuesta, se establece una conexión y el estado se vuelve *ESTABLISHED*. La liberación de la conexión puede iniciarse desde cualquiera de los dos lados. Al completarse, el estado regresa a *CLOSED.*

La máquina de estados finitos se muestra en la figura 6-28. El caso común de un cliente que se conecta activamente a un servidor pasivo se indica con líneas gruesas (continuas para el cliente, punteadas para el servidor). Las líneas delgadas son secuencia de eventos poco comunes. Cada línea de la figura 6-33 se marca mediante un par *evento/acción*. El evento puede ser una llamada de sistema iniciada por el usuario (CONNECT, LISTEN, SEND o CLOSE), la llegada de un segmento (*SYN, FIN, ACK* o *RST*) o, en un caso, una expiración de temporizador del doble del tiempo de vida máximo del paquete. La acción es el envío de un segmento de control (*SYN, FIN* o *RST*), o nada, indicado por —. Los comentarios aparecen entre paréntesis.

Estado	Descripción
CLOSED	No hay conexión activa ni pendiente
LISTEN	El servidor espera una llamada
SYN RCVD	Llegó solicitud de conexión; espera ACK
SYN SENT	La aplicación comenzó a abrir una conexión
ESTABLISHED	Estado normal de transferencia de datos
FIN WAIT 1	La aplicación dijo que ya terminó
FIN WAIT 2	El otro lado acordó liberar
TIMED WAIT	Espera que todos los paquetes mueran
CLOSING	Ambos lados intentaron cerrar simultáneamente
CLOSE WAIT	El otro lado inició una liberación
LAST ACK	Espera que todos los paquetes mueran

Figura 6-32. Estados usados en la máquina de estados finitos de administración de conexiones TCP.

El diagrama puede entenderse mejor siguiendo primero la trayectoria de un cliente (la línea continua gruesa) y luego la de un servidor (línea punteada gruesa). Al emitir una solicitud CONNECT una aplicación de la máquina cliente, la entidad TCP local crea un registro de conexión, lo marca para indicar que está en el estado *SYN SENT,* y envía un segmento *SYN*. Observe que muchas conexiones pueden estar abiertas (o en proceso de apertura) al mismo tiempo como parte de varias aplicaciones, por lo que el estado es por conexión y se asienta en el registro de conexiones. Al llegar el *SYN+ACK,* el TCP envía al *ACK* final del acuerdo de tres vías y se conmuta al estado *ESTABLISHED.* Ahora pueden enviarse y recibirse datos.

Al terminar una aplicación, ejecuta una primitiva CLOSE, que causa que la entidad TCP local envíe un segmento *FIN* y espere el *ACK* correspondiente (recuadro punteado rotulado “cierre activo”). Al llegar el *ACK*, se hace una transición al estado *FIN WAIT* 2, y ya está cerrado un sentido de la conexión. Cuando también cierra el otro lado, llega un *FIN*, para el cual se envía una confirmación de recepción. Ahora ambos lados están cerrados, pero el TCP espera un tiempo igual al tiempo de vida máximo del paquete para garantizar que todos los paquetes de la conexión han sido eliminados, como protección en caso de la pérdida de una confirmación de recepción. Al expirar el temporizador, el TCP borra el registro de la conexión.

Examinemos ahora la administración de la conexión desde el punto de vista del servidor. El servidor hace un LISTEN y se detiene a esperar la aparición de alguien. Al llegar un *SYN*, se envía una confirmación de recepción y el servidor pasa al estado *SYN RCVD*. Cuando llega la confirmación de recepción del *SYN* del servidor, el acuerdo de tres vías se ha completado y el servidor regresa al estado *ESTABLISHED.* Ahora puede ocurrir la transferencia de datos.

Cuando el cliente ha tenido suficiente, hace un CLOSE, que causa la llegada de un *FIN* al servidor (recuadro punteado rotulado “cierre pasivo”). Entonces se envía una señal al servidor. Cuando éste también hace un CLOSE, se envía un *FIN* al cliente. Al llegar la confirmación de recepción del cliente, el servidor libera la conexión y elimina el registro de conexión.

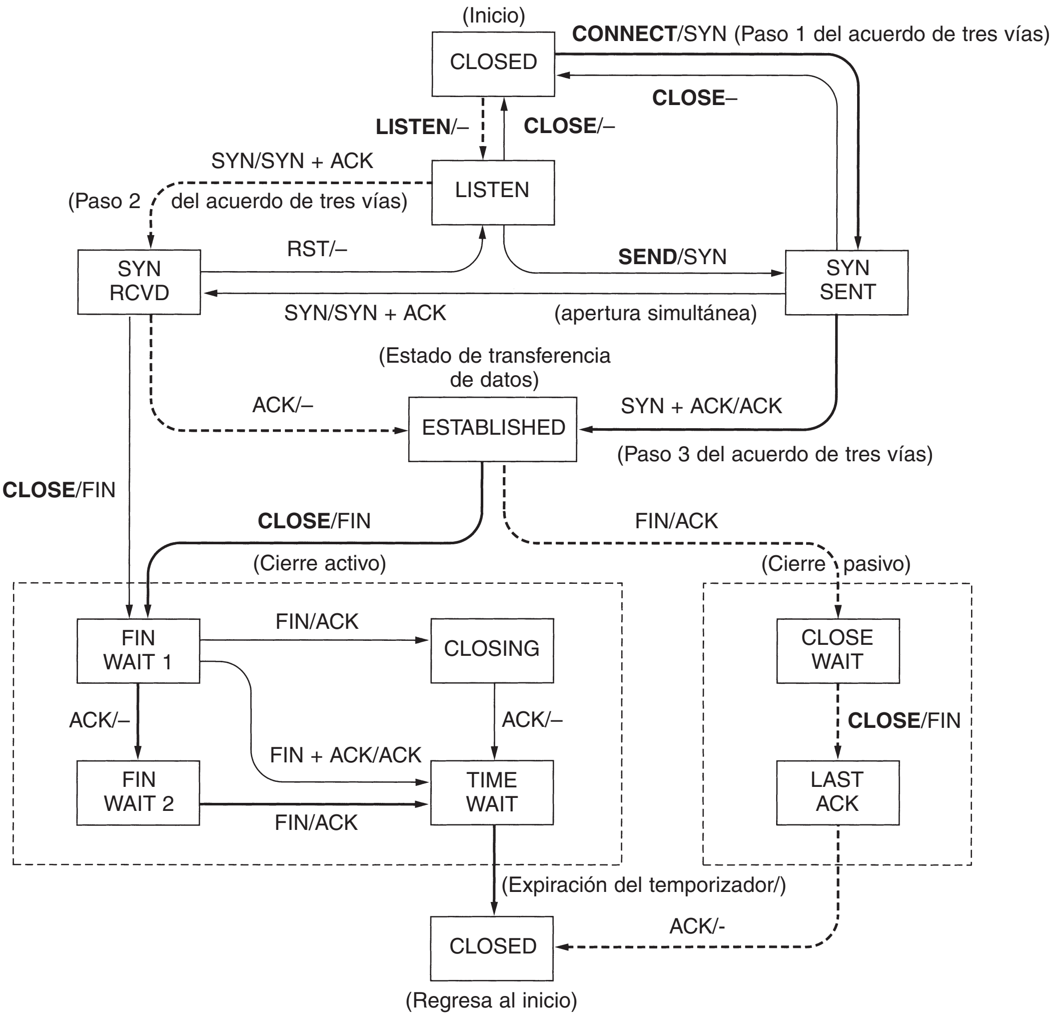


Figura 6-33. Máquina de estados finitos de administración de conexiones TCP. La línea continua gruesa es la trayectoria normal de un cliente. La línea punteada gruesa es la trayectoria normal de un servidor. Las líneas delgadas son eventos poco comunes. Cada transición está indicada por el evento que la ocasiona y la acción resultante, separada por una diagonal.

6.5.8 Política de transmisión del TCP

Como ya vimos, la administración de ventanas en el TCP no está vinculada directamente a las confirmaciones de recepción como en la mayoría de los protocolos de enlace de datos. Por ejemplo, suponga que el receptor tiene un búfer de 4096 bytes, como se muestra en la figura 6-34. Si el emisor envía un segmento de 2048 bytes que se recibe correctamente, el receptor enviará la confirmación de recepción del segmento. Sin embargo, dado que ahora sólo tiene 2048 bytes de

espacio de búfer (hasta que la aplicación retire algunos datos de éste), anunciará una ventana de 2048 comenzando con el siguiente byte esperado.

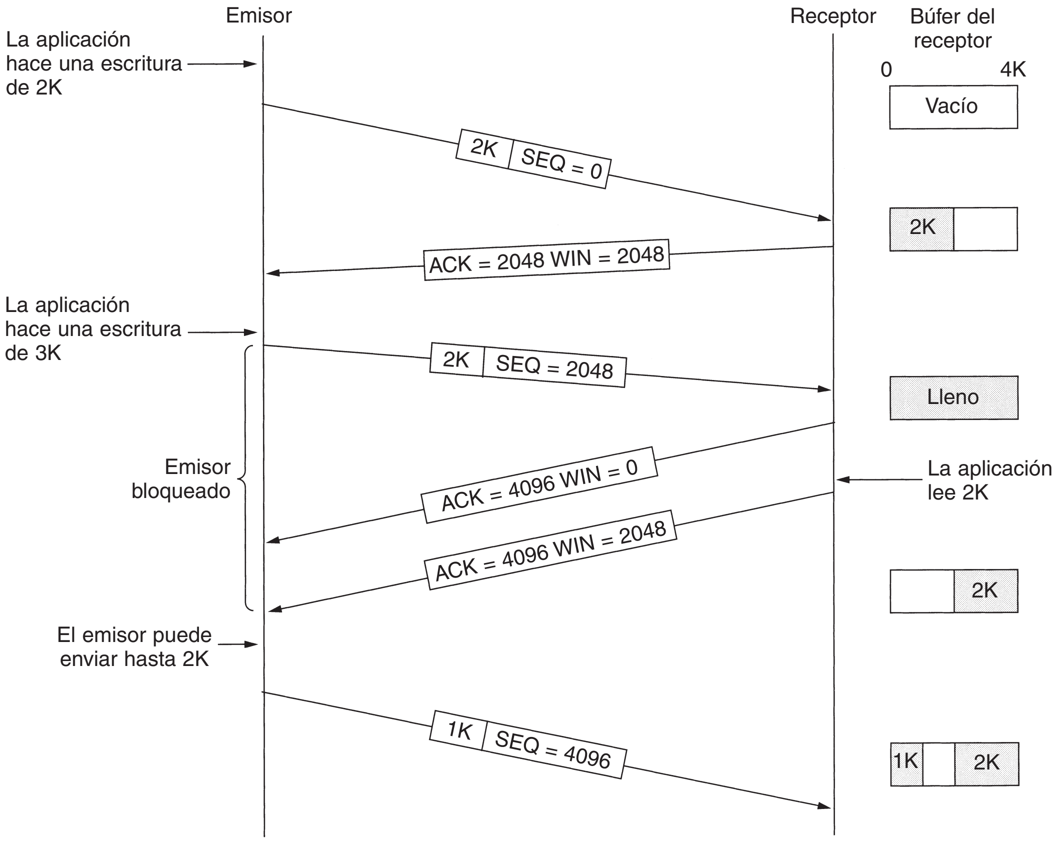


Figura 6-34. Administración de ventanas en TCP.

Ahora el emisor envía otros 2048 bytes, para los cuales el receptor envía la confirmación de recepción, pero la ventana anunciada es de 0. El emisor debe detenerse hasta que el proceso de aplicación del *host* receptor retire algunos datos del búfer, en cuyo momento el TCP puede anunciar una ventana más grande.

Cuando la ventana es de 0, el emisor normalmente no puede enviar segmentos, salvo en dos situaciones. Primera, pueden enviarse datos urgentes (por ejemplo, para permitir que el usuario elimine el proceso en ejecución en la máquina remota). Segunda, el emisor puede enviar un segmento de 1 byte para hacer que el receptor reanuncie el siguiente byte esperado y el tamaño de la ventana. El estándar TCP proporciona explícitamente esta opción para evitar un bloqueo irreversible si llega a perderse un anuncio de ventana.

No se requiere que los emisores envíen datos tan pronto como llegan de la aplicación. Tampoco se requiere que los receptores envíen confirmaciones de recepción tan pronto como sea posible. Por ejemplo, en la figura 6-34, cuando llegaron los primeros 2 KB de datos, el TCP, sabiendo que tenía disponible una ventana de 4 KB, habría actuado perfectamente bien si simplemente almacena en búfer los datos hasta la llegada de otros 2 KB, para poder transmitir un segmento con una carga útil de 4 KB. Esta libertad puede explotarse para mejorar el desempeño.

Considere una conexión telnet a un editor interactivo que reacciona con cada pulso de tecla. En el peor caso, al llegar un carácter a la entidad TCP emisora, el TCP crea un segmento TCP de 21 bytes que entrega al IP para su envío como datagrama IP de 41 bytes. Del lado receptor, el TCP envía de inmediato una confirmación de recepción de 40 bytes (20 bytes de encabezado TCP y 20 bytes de encabezado IP). Después, cuando el editor ha leído el byte, el TCP envía una actualización de ventana, recorriendo la ventana 1 byte hacia la derecha. Este paquete también es de 40 bytes. Por último, cuando el editor ha procesado el carácter, lo retransmite como paquete de 41 bytes. En conjunto, se usan 162 bytes de ancho de banda y se envían cuatro segmentos por cada carácter pulsado. Cuando es escaso el ancho de banda, no es deseable este método de operación.

Un enfoque que usan muchas implementaciones del TCP para mejorar esta situación es el retardo de las confirmaciones de recepción y de las actualizaciones de ventana durante 500 mseg con la esperanza de que lleguen algunos datos con los cuales viajar gratuitamente. Suponiendo que el editor hace eco en un lapso de 500 mseg, sólo se necesita enviar ahora un paquete de 41 bytes de regreso al usuario remoto, recortando a la mitad la cuenta de paquetes y el uso de ancho de banda.

Aunque esta regla reduce la carga impuesta a la red por el receptor, éste aún opera de manera ineficiente al enviar paquetes de 41 bytes que contienen 1 byte de datos. Una manera de reducir este uso es empleando el **algoritmo de Nagle** (Nagle, 1984). Lo que sugirió Nagle es sencillo: al llegar datos al emisor un byte a la vez, simplemente se envía el primer byte y se almacena en búfer los demás hasta la confirmación de recepción del byte pendiente. Luego se transmiten todos los caracteres del búfer en un segmento TCP y nuevamente se comienzan a almacenar en búfer los datos hasta que se haya confirmado la recepción de todos. Si el usuario escribe con rapidez y la red es lenta, puede entrar una cantidad importante de caracteres en cada segmento, reduciendo en gran medida el ancho de banda usado. Además, el algoritmo permite el envío de un nuevo paquete si han entrado suficientes datos para llenar la mitad de la ventana o la totalidad de un segmento.

El algoritmo de Nagle se usa ampliamente en las implementaciones de TCP, pero hay veces en que es mejor inhabilitarlo. En particular, al operar una aplicación X-Windows a través de Internet, los movimientos del ratón tienen que enviarse a la computadora remota. (X-Windows es el sistema de ventanas que se utiliza en la mayoría de los sistemas UNIX.) Su acumulación para enviarlos en ráfagas hace que el movimiento del cursor sea errático, lo que no complace mucho a los usuarios.

Otro problema que puede arruinar el desempeño del TCP es el **síndrome de ventana tonta** (Clark, 1982). Este problema ocurre cuando se pasan datos a la entidad emisora en bloques grandes, pero una aplicación interactiva del lado receptor lee datos a razón de 1 byte a la vez. Para ver el problema, observe la figura 6-35. Inicialmente, el búfer TCP del lado receptor está lleno y el

emisor lo sabe (es decir, tiene un tamaño de ventana de 0). Entonces la aplicación interactiva lee un carácter del flujo TCP. Esta acción hace feliz al TCP receptor, por lo que envía una actualización de ventana al emisor indicando que está bien que envíe 1 byte. El emisor accede y envía 1 byte. El búfer ahora está lleno, por lo que el receptor confirma la recepción del segmento de 1 byte pero establece la ventana en 0. Este comportamiento puede continuar indefinidamente.

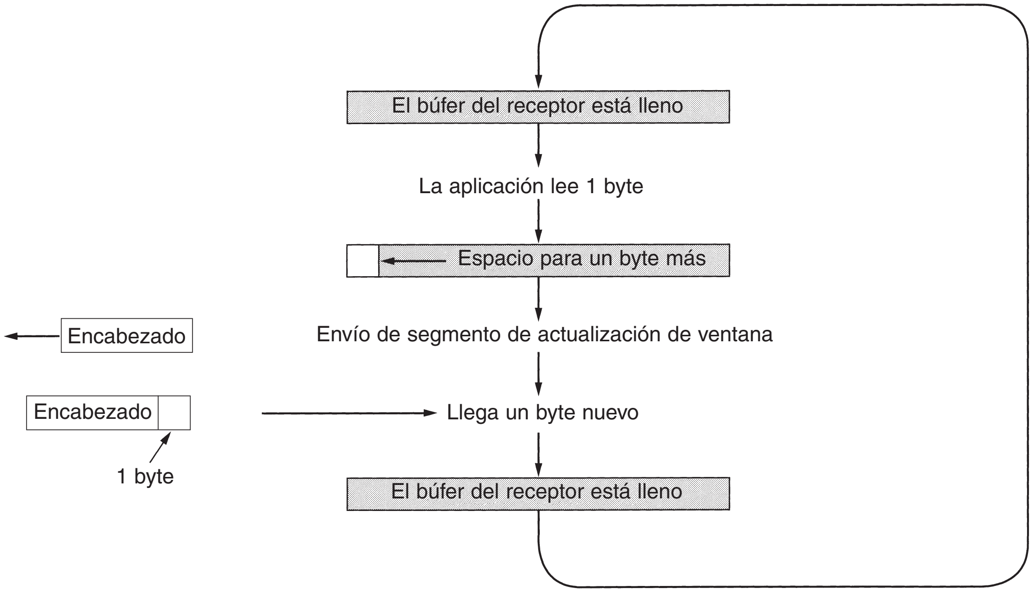


Figura 6-35. Síndrome de ventana tonta.

La solución de Clark es evitar que el receptor envíe una actualización de ventana para 1 byte. En cambio, se le obliga a esperar hasta tener disponible una cantidad de espacio, y luego lo anuncia. Específicamente, el receptor no debe enviar una actualización de ventana hasta que pueda manejar el tamaño máximo de segmento que anunció al establecerse la conexión, o que su búfer quede a la mitad de capacidad, lo que sea más pequeño.

Además, el emisor también puede ayudar al no enviar segmentos muy pequeños. En cambio, debe intentar esperar hasta haber acumulado suficiente espacio en la ventana para enviar un segmento completo, o cuando menos uno que contenga la mitad del tamaño de búfer del receptor (que debe estimar a partir del patrón de las actualizaciones de ventana que ha recibido anteriormente).

El algoritmo de Nagle y la solución de Clark al síndrome de ventana tonta son complementarios. Nagle trataba de resolver el problema causado por la entrega de datos al TCP desde la aplicación emisora un byte a la vez. Clark trataba de resolver el problema de que la aplicación receptora toma los datos del TCP un byte a la vez. Ambas soluciones son válidas y pueden operar juntas. La meta es que el emisor no envíe segmentos pequeños y que el receptor no los pida.

El TCP receptor también puede hacer más para mejorar el desempeño que simplemente actualizar ventanas en unidades grandes. Al igual que el TCP emisor, tiene la capacidad de almacenar datos en el búfer, por lo que puede bloquear una solicitud READ de la aplicación hasta poder proporcionar un bloque grande de datos. Hacer esto reduce la cantidad de llamadas al TCP y, por tanto, la sobrecarga. Por supuesto, también aumenta el tiempo de respuesta, pero en las aplicaciones no interactivas, como la transferencia de archivos, la eficiencia puede tener mayor peso que el tiempo de respuesta a las solicitudes individuales.

Otro problema del receptor es qué debe hacer con los segmentos fuera de orden. Pueden conservarse o descartarse, al albedrío del receptor. Por supuesto, las confirmaciones de recepción pueden enviarse sólo después de haber recibido todos los datos hasta el byte confirmado. Si el receptor recibe los segmentos 0, 1, 2, 4, 5, 6 y 7, puede enviar una confirmación de recepción de todos los bytes hasta el último byte del segmento 2, inclusive. Al expirar el temporizador del emisor, retransmitirá el segmento 3. Si el receptor tienen en búfer los segmentos 4 a 7, al recibir el segmento 3 puede enviar una confirmación de recepción de todos los bytes hasta el final del segmento 7.

6.5.9 Control de congestión en TCP

Cuando la carga ofrecida a cualquier red es mayor que la que puede manejar, se genera una congestión. Internet no es ninguna excepción. En esta sección estudiaremos los algoritmos que se han desarrollado durante la última década para manejar la congestión. Aunque la capa de red también intenta manejarlos, gran parte del trabajo pesado recae sobre el TCP porque la solución real a la congestión es la disminución de la tasa de datos.

En teoría, puede manejarse la congestión aprovechando un principio de física: la ley de conservación de los paquetes. La idea es no inyectar un paquete nuevo en la red hasta que salga uno viejo (es decir, se entregue). El TCP intenta lograr esta meta manipulando dinámicamente el tamaño de la ventana.

El primer paso del manejo de la congestión es su detección. Antaño la detección de congestionamientos era muy difícil. La expiración de un temporizador causada por un paquete perdido podía deberse a (1) ruido en la línea de transmisión o (2) el descarte de paquetes en el enrutador congestionado. Saber cuál era la diferencia era difícil.

Hoy día, la pérdida de paquetes por errores de transmisión es relativamente rara debido a que las troncales de larga distancia son de fibra (aunque las redes inalámbricas son otra historia). En consecuencia, la mayoría de las expiraciones de tiempo en Internet se deben a congestión. Todos los algoritmos TCP de Internet suponen que las expiraciones de tiempo son causadas por congestión y las revisan en busca de problemas, de la misma manera que los mineros observan a sus canarios.

Antes de analizar la manera en que el TCP reacciona a la congestión, describiremos primero lo que hace para evitar que ocurra. Al establecerse una conexión, se tiene que seleccionar un tamaño de ventana adecuado. El receptor puede especificar una ventana con base en su tamaño de búfer. Si el emisor se ajusta a su tamaño de ventana, no ocurrirán problemas por desbordamiento de búferes en la terminal receptora, pero aún pueden ocurrir debido a congestión interna en la red.

En la figura 6-36 ilustramos este problema hidráulicamente. En la figura 6-36(a) vemos un tubo grueso que conduce a un receptor de poca capacidad. Mientras el emisor no envíe más agua de la que puede contener la cubeta, no se perderá agua. En la figura 6-36(b), el factor limitante no es la capacidad de la cubeta, sino la capacidad de conducción interna de la red. Si entra demasiada agua a alta velocidad, ésta retrocederá, perdiéndose algo (en este caso, por el desbordamiento del embudo).

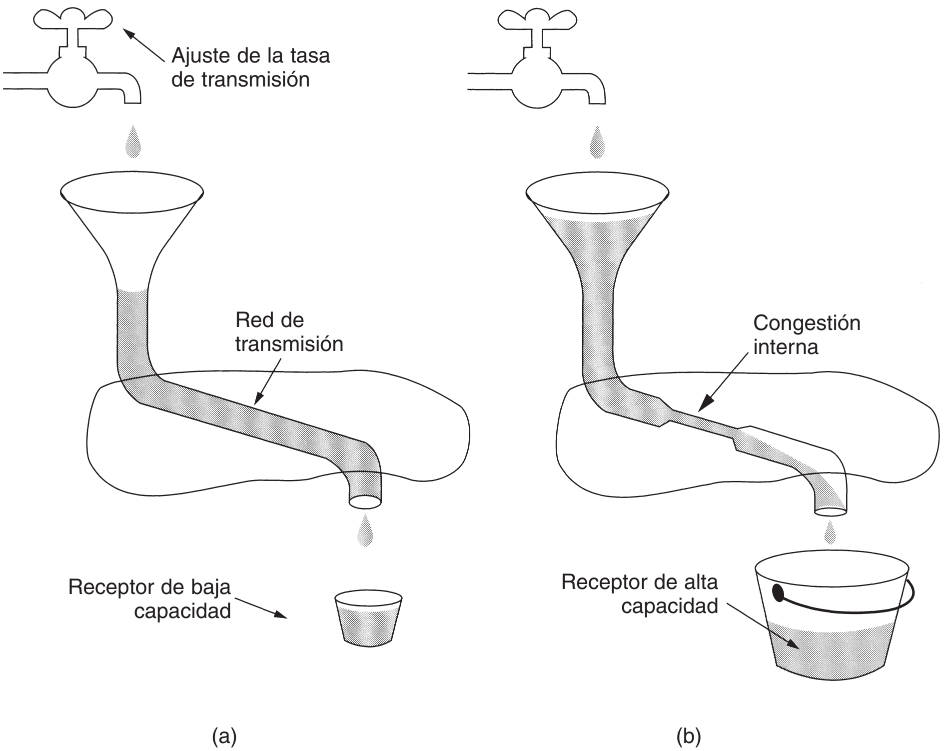


Figura 6-36. (a) Red rápida alimentando un receptor de baja capacidad. (b) Red lenta alimentando un receptor de alta capacidad.

La solución de Internet es aceptar que existen dos problemas potenciales (capacidad de la red y capacidad del receptor) y manejarlos por separado. Para ello, cada emisor mantiene dos ventanas: la ventana que ha otorgado el receptor y una segunda ventana, la **ventana de congestión.** Cada una refleja la cantidad de bytes que puede enviar el emisor. La cantidad de bytes que pueden enviarse es la cifra menor de las dos ventanas. Por tanto, la ventana efectiva es el mínimo de lo que el emisor piensa que es correcto y lo que el receptor piensa que está bien. Si el receptor dice "envía 8 KB" pero el emisor sabe que las ráfagas de más de 4 KB saturan la red, envía 4 KB. Por otra parte, si el receptor dice "envía 8 KB" y el emisor sabe que las ráfagas de hasta 32 KB pueden llegar sin problemas, envía los 8 KB solicitados.

Al establecer una conexión, el emisor asigna a la ventana de congestión el tamaño de segmento máximo usado por la conexión; entonces envía un segmento máximo. Si se recibe la confirmación de recepción de este segmento antes de que expire el temporizador, el emisor agrega el equivalente en bytes de un segmento a la ventana de congestión para hacerla de dos segmentos de tamaño máximo, y envía dos segmentos. A medida que se confirma cada uno de estos segmentos, se aumenta el tamaño de la ventana de congestión en un segmento máximo. Cuando la ventana de congestión es de *n* segmentos, si de todos los *n* se reciben confirmaciones de recepción a tiempo, se aumenta el tamaño de la ventana de congestión en la cuenta de bytes correspondiente a *n* segmentos. De hecho, cada ráfaga confirmada duplica la ventana de congestionamiento.

La ventana de congestión sigue creciendo exponencialmente hasta ocurrir una expiración del temporizador o alcanzar el tamaño de la ventana receptora. La idea es que, si las ráfagas de 1024, 2048 y 4096 bytes funcionan bien, pero una ráfaga de 8192 produce una expiración del temporizador, la ventana de congestión debe establecerse en 4096 para evitar la congestión. Mientras el tamaño de la ventana de congestión permanezca en 4096, no se enviará una ráfaga de mayor longitud, sin importar la cantidad de espacio de ventana otorgada por el receptor. Este algoritmo se llama **arranque lento,** pero no es lento en lo absoluto (Jacobson, 1988); es exponencial, y se requiere que todas las implementaciones de TCP lo manejen.

Veamos ahora el algoritmo de control de congestión de Internet, el cual usa un tercer parámetro, el **umbral,** inicialmente de 64 KB, además de las ventanas de recepción y congestión. Al ocurrir una expiración del temporizador, se establece el umbral en la mitad de la ventana de congestión actual, y la ventana de congestión se restablece a un segmento máximo. Luego se usa el arranque lento para determinar lo que puede manejar la red, excepto que el crecimiento exponencial termina al alcanzar el umbral. A partir de este punto, las transmisiones exitosas aumentan linealmente la ventana de congestión (en un segmento máximo por ráfaga) en lugar de uno por segmento. En efecto, este algoritmo está suponiendo que probablemente es aceptable recortar la ventana de congestión a la mitad, y luego aumentarla gradualmente a partir de ahí.

Como ilustración de la operación del algoritmo de congestión, véase la figura 6-37. El tamaño máximo de segmento aquí es de 1024 bytes. Inicialmente, la ventana de congestión era de 64 KB, pero ocurre una expiración del temporizador, así que se establece el umbral en 32KB y la ventana de congestión en 1KB, para la transmisión 0. La ventana de congestión entonces crece exponencialmente hasta alcanzar el umbral (32 KB). A partir de entonces, crece linealmente.

La transmisión 13 tiene mala suerte (debería saberlo) y ocurre una expiración del temporizador. Se establece el umbral en la mitad de la ventana actual (ahora de 40 KB, por lo que la mitad es de 20 KB), e inicia de nuevo el arranque lento. Al llegar las confirmaciones de recepción de la transmisión 14, los primeros cuatro incrementan la ventana de congestión en un segmento máximo, pero después de eso el crecimiento se vuelve lineal nuevamente.

Si no ocurren más expiraciones del temporizador, la ventana de congestión continuará creciendo hasta el tamaño de la ventana del receptor. En ese punto, dejará de crecer y permanecerá constante mientras no ocurran más expiraciones del temporizador y la ventana del receptor no cambie de tamaño. Como nota al margen, si llega un paquete SOURCE QUENCH de ICMP y pasa al TCP, este evento será tratado de la misma manera que una expiración del temporizador. Un enfoque alternativo (y más reciente) se describe en el RFC 3168.

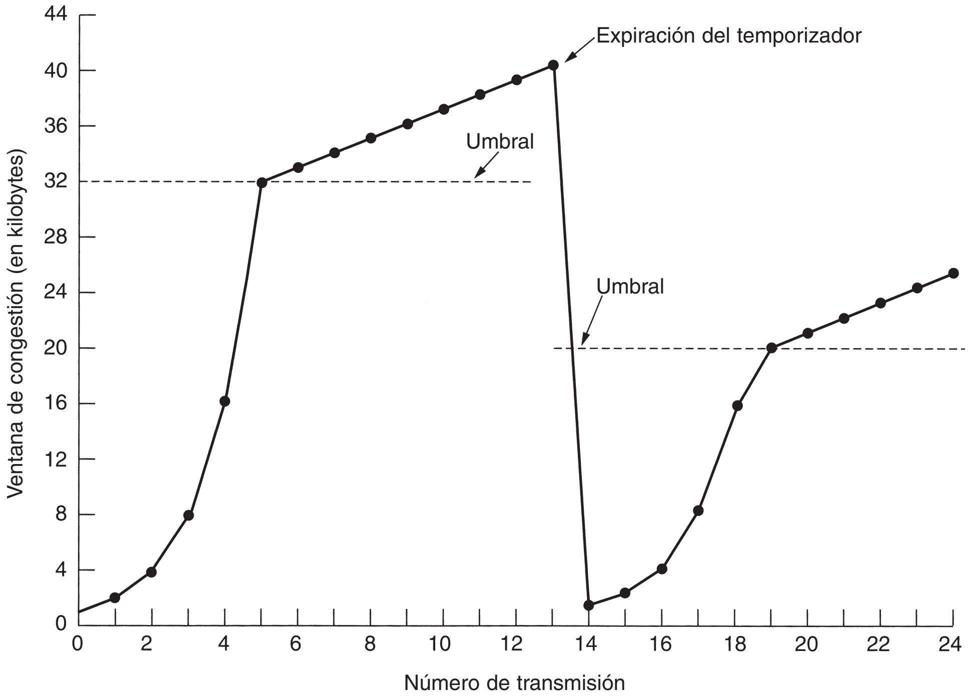


Figura 6-37. Ejemplo del algoritmo de congestión de Internet.

6.5.10 Administración de temporizadores del TCP

El TCP usa varios temporizadores (al menos conceptualmente) para hacer su trabajo. El más importante de éstos es el **temporizador de retransmisión.** Al enviarse un segmento, se inicia un temporizador de retransmisiones. Si la confirmación de recepción del segmento llega antes de expirar el temporizador, éste se detiene. Si, por otra parte, el temporizador termina antes de llegar la confirmación de recepción, se retransmite el segmento (y se inicia nuevamente el temporizador). Surge entonces la pregunta: ¿qué tan grande debe ser el intervalo de expiración del temporizador?

Este problema es mucho más difícil en la capa de transporte de Internet que en los protocolos de enlace de datos genéricos del capítulo 3. En este último caso, el retardo esperado es altamente predecible (es decir, tiene una variación baja), por lo que el temporizador puede ejecutarse para expirar justo después del momento en que se espera la confirmación de recepción, como se muestra en la figura 6-38(a). Dado que las confirmaciones de recepción pocas veces se retardan en la capa de enlace de datos (debido a la falta de congestión), la ausencia de una confirmación de recepción en el momento esperado generalmente significa que la trama o la confirmación de recepción se han perdido.

El TCP enfrenta un entorno radicalmente distinto. La función de densidad de probabilidad del tiempo que tarda en regresar una confirmación de recepción TCP se parece más a la figura 6-38(b) que a la figura 6-38(a). Es complicada la determinación del tiempo de ida y vuelta al destino. Aun

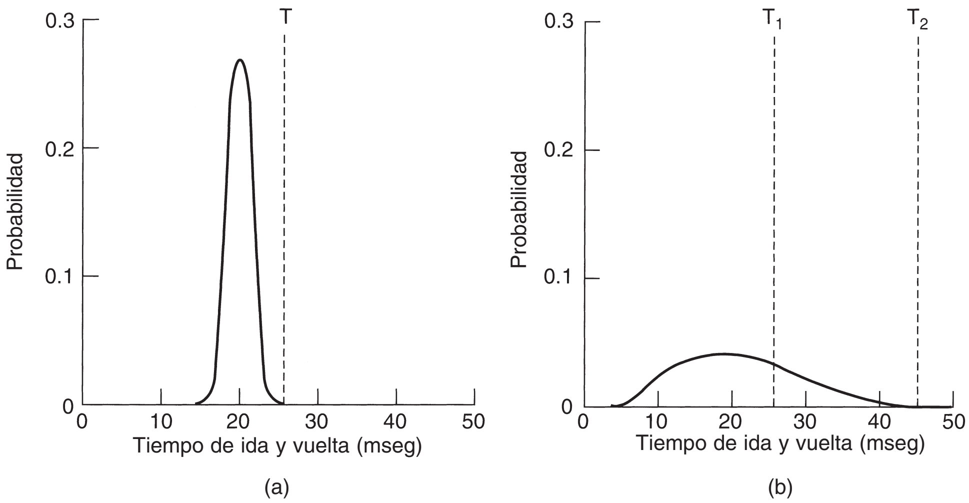


Figura 6-38. (a) Densidad de probabilidad de los tiempos de llegada de confirmaciones de recepción en la capa de enlace de datos. (b) Densidad de probabilidad de los tiempos de llegada de confirmaciones de recepción para el TCP.

cuando se conoce, la selección del intervalo de expiración del temporizador también es difícil. Si se hace demasiado corto, digamos T_1 en la figura 6-38(b), ocurrirán retransmisiones innecesarias, cargando la Internet con paquetes inútiles. Si se hace demasiado largo (por ejemplo, T_2), el desempeño sufrirá debido al gran retardo de retransmisión de cada paquete perdido. Es más, la varianza y la media de la distribución de llegadas de confirmaciones de recepción pueden variar con rapidez en unos cuantos segundos, a medida que se generan y se resuelven congestionamientos.

La solución es usar un algoritmo muy dinámico que ajuste constantemente el intervalo de expiración del temporizador, con base en mediciones continuas del desempeño de la red. El algoritmo que generalmente usa el TCP lo debemos a Jacobson (1988) y funciona como sigue. Por cada conexión, el TCP mantiene una variable, *RTT* (*round-trip time*), que es la mejor estimación actual del tiempo de ida y vuelta al destino en cuestión. Al enviarse un segmento, se inicia un temporizador, tanto para ver el tiempo que tarda la confirmación de recepción como para habilitar una retransmisión si se tarda demasiado. Si llega la confirmación de recepción antes de expirar el temporizador, el TCP mide el tiempo que tardó la confirmación de recepción, digamos M. Entonces actualiza *RTT* de acuerdo con la fórmula

$$RTT = \alpha RTT + (1 - \alpha)M$$

donde α es un factor de amortiguamiento que determina el peso que se le da al valor anterior. Por lo común, $\alpha = 7/8$.

Aun dado un buen valor de *RTT*, la selección de una expiración adecuada del temporizador de retransmisión no es un asunto sencillo. Normalmente el TCP usa βRTT, pero el truco es seleccionar β. En las implementaciones iniciales, β siempre era 2, pero la experiencia demostró que un valor constante era inflexible puesto que no respondía cuando subía la variación.

En 1988, Jacobson propuso hacer que β fuera aproximadamente proporcional a la desviación estándar de la función de densidad de probabilidad del tiempo de llegada de las confirmaciones de recepción, por lo que una variación grande significa una β grande, y viceversa. En particular, sugirió el uso de la *desviación media* como una forma rápida de estimar la *desviación estándar.* Su algoritmo requiere mantener otra variable amortiguada, *D,* la desviación. Al llegar una confirmación de recepción, se calcula la diferencia entre el valor esperado y el observado, $|RTT - M|$. Un valor amortiguado de esta cifra se mantiene en *D* mediante la fórmula

$$D = \alpha D + (1 - \alpha)\ |RTT - M|$$

donde α puede ser o no el mismo valor usado para amortiguar *RTT*. Si bien *D* no es exactamente igual a la desviación estándar, es bastante buena, y Jacobson demostró la manera de calcularla usando sólo sumas, restas y desplazamientos de enteros, lo que es una gran ventaja. La mayoría de las implementaciones TCP usan ahora este algoritmo y establecen el intervalo de expiración del temporizador en

$$\text{Expiración del temporizador} = RTT + 4 \times D$$

La selección del factor 4 es un tanto arbitraria, pero tiene dos ventajas. Primera, puede hacerse la multiplicación por 4 con un solo desplazamiento. Segunda, reduce al mínimo las expiraciones de temporizador y las retransmisiones innecesarias porque menos del 1% de todos los paquetes llegan más de cuatro desviaciones estándar tarde. (En realidad, Jacobson sugirió inicialmente que se usaran 2, pero el trabajo posterior ha demostrado que 4 da un mejor desempeño.)

Un problema que ocurre con la estimación dinámica de *RTT* es qué se debe hacer cuando expira el temporizador de un segmento y se envía de nuevo. Cuando llega la confirmación de recepción, no es claro si éste se refiere a la primera transmisión o a una posterior. Si se adivina mal se puede contaminar seriamente la estimación de *RTT*. Phil Karn descubrió este problema de la manera difícil. Él es un radioaficionado interesado en la transmisión de paquetes TCP/IP a través de la radio amateur, un medio notoriamente poco confiable (en un buen día, pasarán la mitad de los paquetes). Karn hizo una propuesta sencilla: no actualizar el *RTT* con ninguno de los segmentos retransmitidos. En cambio, se duplica la expiración del temporizador con cada falla hasta que los segmentos pasan a la primera. Este sistema se llama **algoritmo de Karn** y lo usan la mayoría de las implementaciones TCP.

El temporizador de retransmisiones no es el único usado por el TCP. El segundo temporizador es el **temporizador de persistencia,** diseñado para evitar el siguiente bloqueo irreversible. El receptor envía una confirmación de recepción con un tamaño de ventana de 0, indicando al emisor que espere. Después, el receptor actualiza la ventana, pero se pierde al paquete con la actualización. Ahora, tanto el emisor como el receptor están esperando que el otro haga algo. Cuando termina el temporizador de persistencia, el emisor envía un sondeo al receptor. La respuesta al sondeo da el tamaño de la ventana. Si aún es de cero, se inicia el temporizador de persistencia nuevamente y se repite el ciclo. Si es diferente de cero, pueden enviarse datos.

Un tercer temporizador usado en algunas implementaciones es el **temporizador de seguir con vida** (*keepalive timer*). Cuando una conexión ha estado inactiva durante demasiado tiempo, el

temporizador de seguir con vida puede expirar, haciendo que un lado compruebe que el otro aún está ahí. Si no se recibe respuesta, se termina la conexión. Esta característica es motivo de controversias porque agrega sobrecarga y puede terminar una conexión saludable debido a una partición temporal de la red.

El último temporizador usado en cada conexión TCP es el que se usa en el estado *TIMED WAIT* durante el cierre; opera durante el doble del tiempo máximo de vida de paquete para asegurar que, al cerrarse una conexión, todos los paquetes creados por ella hayan desaparecido.

6.5.11 TCP y UDP inalámbricos

En teoría, los protocolos de transporte deben ser independientes de la tecnología de la capa de red subyacente. En particular, el TCP no debería preocuparse si el IP está operando por fibra o por radio. En la práctica sí importa, puesto que la mayoría de las implementaciones de TCP han sido optimizadas cuidadosamente con base en supuestos que se cumplen en las redes alámbricas, pero no en las inalámbricas. Ignorar las propiedades de la transmisión inalámbrica puede conducir a implementaciones del TCP correctas desde el punto de vista lógico pero con un desempeño horrendo.

El problema principal es el algoritmo de control de congestionamientos. Hoy día, casi todas las implementaciones de TCP suponen que las expiraciones del temporizador ocurren por congestionamientos, no por paquetes perdidos. En consecuencia, al expirar un temporizador, el TCP disminuye su velocidad y envía con menor ímpetu (por ejemplo, el algoritmo de arranque lento de Jacobson). Lo que se pretende con este enfoque es reducir la carga de la red y aliviar así la congestión.

Desgraciadamente, los enlaces de transmisión inalámbrica son muy poco confiables. Pierden paquetes todo el tiempo. El enfoque adecuado para el manejo de paquetes perdidos es enviarlos nuevamente, tan pronto como sea posible. La reducción de la velocidad simplemente empeora las cosas. Si, digamos, se pierde el 20% de todos los paquetes, entonces cuando el emisor envía 100 paquetes/seg, la velocidad real de transporte es de 80 paquetes/seg. Si el emisor reduce su velocidad a 50 paquetes/seg, la velocidad real de transporte cae a 40 paquetes/seg.

En efecto, al perderse un paquete en una red alámbrica, el emisor debe reducir la velocidad. Cuando se pierde uno en una red inalámbrica, el emisor debe acelerar. Cuando el emisor no sabe de qué clase de red se trata, es difícil tomar la decisión correcta.

Con frecuencia, la trayectoria del emisor al receptor no es homogénea. Los primeros 1000 km podrían ser a través de una red alámbrica, pero el último km podría ser inalámbrico. Ahora es más difícil la decisión correcta en el caso de una expiración del temporizador, ya que es importante saber dónde ocurrió el problema. Una solución propuesta por Bakne y Badrinath (1995), el **TCP indirecto,** es la división de la conexión TCP en dos conexiones distintas, como se muestra en la figura 6-39. La primera va del emisor a la estación base. La segunda va de la estación base al receptor. La estación base simplemente copia paquetes entre las conexiones en ambas direcciones.

La ventaja de este esquema es que ahora ambas conexiones son homogéneas. Las expiraciones del temporizador en la primera conexión pueden reducir la velocidad del emisor, y las expiraciones del temporizador en la segunda pueden acelerarla. También es posible ajustar otros parámetros

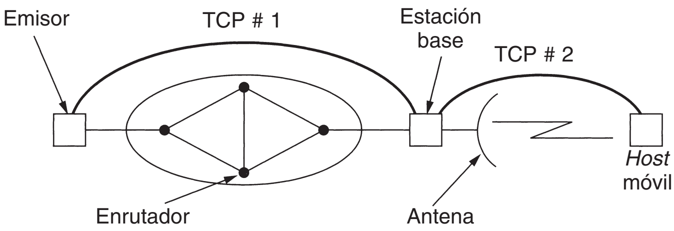


Figura 6-39. División de una conexión TCP en dos conexiones.

por separado para cada conexión. La desventaja es que se viola por completo la semántica del TCP. Dado que cada parte de la conexión es una conexión TCP completa, la estación base confirma la recepción de cada segmento TCP de la manera normal, sólo que ahora la recepción de una confirmación en el emisor no significa que el receptor recibió el segmento, sino que la estación base lo recibió.

Una solución diferente, debido a Balakrishnan y cols. (1995), no quebranta la semántica del TCP. Funciona haciendo varias modificaciones pequeñas al código de la capa de red de la estación base. Uno de los cambios es la adición de un agente espía que observa y almacena en caché los segmentos TCP que van al *host* móvil y las confirmaciones de recepción que regresan de él. Cuando el agente espía ve un segmento TCP que sale al *host* móvil, pero no ve el regreso de una confirmación de recepción antes de que su temporizador (relativamente corto) expire, simplemente retransmite ese segmento, sin indicar al origen que lo está haciendo. El agente también genera una retransmisión cuando detecta confirmaciones de recepción duplicadas del *host* móvil, lo que invariablemente significa que algo le ha fallado a este *host*. Las confirmaciones de recepción duplicadas se descartan en seguida, para evitar que el origen las malinterprete como una señal de congestionamiento.

Sin embargo, una desventaja de esta transparencia es que, si el enlace inalámbrico tiene muchas pérdidas, el temporizador del transmisor podría expirar esperando una confirmación de recepción e invocar el algoritmo de control de congestión. Con en TCP indirecto, el algoritmo de control de congestión nunca iniciará hasta que realmente haya congestión en la parte alámbrica de la red.

El documento de Balakrishnan y cols., también sugiere una solución al problema de segmentos perdidos que se originan en el *host* móvil. Al notar una estación base un hueco en los números de secuencia de entrada, genera una solicitud de repetición selectiva de los bytes faltantes usando una opción TCP.

Gracias a estos dos mecanismos, el enlace inalámbrico se hace más confiable en ambas direcciones, sin que el origen lo sepa y sin cambiar la semántica del TCP.

Si bien el UDP no tiene los mismos problemas que el TCP, la comunicación inalámbrica también le produce dificultades. El problema principal es que los programas usan el UDP pensando que es altamente confiable. Saben que no hay garantías, pero aun así esperan que sea casi perfecto. En un entorno inalámbrico, UDP estará muy lejos de serlo. En aquellos programas capaces de recuperarse de la pérdida de mensajes UDP, pasar repentinamente de un entorno en el que pueden

perderse mensajes, pero rara vez ocurre, a uno en el que se pierden constantemente, puede dar pie a un desempeño desastroso.

La comunicación inalámbrica también afecta otras áreas, no sólo el desempeño. Por ejemplo, ¿cómo encuentra un *host* móvil una impresora local a la cual conectarse, en lugar de usar su impresora base? Algo parecido a esto es cómo acceder a la página WWW de la celda local, aun si no se conoce su nombre. También, los diseñadores de páginas WWW tienden a suponer que hay mucho ancho de banda disponible. Un logotipo grande en cada página se vuelve contraproducente si su transmisión tarda 10 seg en un enlace inalámbrico lento cada vez que se hace referencia a la página, irritando sobremanera a los usuarios.

Conforme las redes inalámbricas se vuelvan más comunes, los problemas de ejecutar TCP sobre ellas se volverán más serios. En (Barakat y cols., 2000; Ghani y Dixit, 1999; Huston, 2001, y Xylomenos y cols., 2001), encontrará información adicional sobre esta área.

6.5.12 TCP para Transacciones

Al inicio de este capítulo vimos las llamadas a procedimiento remoto como una forma de implementar sistemas cliente-servidor. Si tanto la solicitud como la respuesta son suficientemente pequeñas para caber en paquetes sencillos y la operación tiene la misma potencia, simplemente se puede utilizar UDP. Sin embargo, si estas condiciones no se cumplen, el uso de UDP no es tan conveniente. Por ejemplo, si la respuesta puede ser más grande, las piezas deben seguir una secuencia y se debe diseñar un mecanismo para retransmitir las piezas perdidas. En efecto, la aplicación tiene que remodelar el TCP.

Es obvio que esto no resulta tan conveniente, pero tampoco lo es utilizar el TCP mismo. El problema es la eficiencia. En la figura 6-40(a) se muestra la secuencia normal de paquetes para realizar una RPC en TCP. En el mejor de los casos se necesitan los siguientes nueve paquetes.

1. El cliente envía un paquete *SYN* para establecer una conexión.
2. El servidor envía un paquete *ACK* para confirmar la recepción del paquete *SYN*.
3. El cliente completa el acuerdo de tres vías.
4. El cliente envía la solicitud real.
5. El cliente envía un paquete *FIN* para indicar que ha terminado el envío.
6. El servidor confirma la recepción de la solicitud y el paquete *FIN*.
7. El servidor envía la respuesta al cliente.
8. El servidor envía un paquete *FIN* para indicar que también ha terminado.
9. El cliente confirma la recepción del paquete *FIN* del servidor.

Observe que éste es el mejor de los casos. En el peor, la confirmación de recepción de la solicitud del cliente y del paquete *FIN* se realiza por separado, al igual que la respuesta y el paquete *FIN* del servidor.

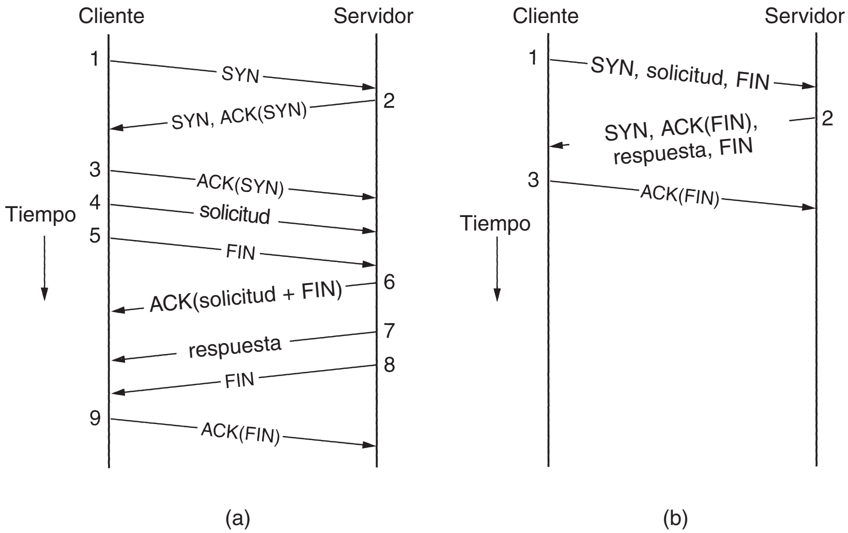


Figura 6-40. (a) RPC mediante el TCP normal. (b) RPC mediante el T/TCP.

Con esto surge rápidamente la pregunta de si hay alguna forma para combinar la eficiencia de RPC utilizando UDP (sólo dos mensajes) con la confiabilidad de TCP. La respuesta es: Casi. Puede hacerse con una variante TCP experimental llamada **T/TCP** (**TCP para Transacciones**), que se describe en los RFCs 1379 y 1644.

La idea central es modificar ligeramente la secuencia estándar de configuración de conexión para permitir la transferencia de datos durante la configuración. En la figura 6-40(b) se ilustra el protocolo T/TCP. El primer paquete del cliente contiene el bit *SYN*, la solicitud misma y el paquete *FIN*. Lo que dice es: Deseo establecer una conexión, aquí están los datos, y con esto termino.

Cuando el servidor obtiene la solicitud, busca o calcula la respuesta y elige cómo responder. Si la respuesta se ajusta en un paquete, da la respuesta de la figura 6-40(b), que dice: Confirmo la recepción de tu paquete *FIN*, aquí está la respuesta, y con esto termino. A continuación el cliente confirma la recepción del paquete *FIN* del servidor, y el protocolo termina en tres mensajes.

Sin embargo, si el resultado es de más de un paquete, el servidor también tiene la opción de no encender el bit *FIN*, en cuyo caso puede enviar múltiples paquetes antes de cerrar su dirección.

Probablemente valga la pena mencionar que T/TCP no es la única mejora propuesta a TCP. Otra propuesta es **SCTP** (**Protocolo de Transmisión de Control de Flujo**). Sus características incluyen preservación de los límites de mensajes, modos de entrega múltiples (por ejemplo, entrega en desorden), multihoming (destinos de respaldo) y confirmaciones de recepción selectivas (Stewart y Metz, 2001). Sin embargo, siempre que alguien propone cambiar algo que ha trabajado bien por algún tiempo considerable, hay una gran batalla entre las siguientes posturas: “Los usuarios están demandando más características” y “Si no está roto, no lo arregles”.

6.6 ASPECTOS DEL DESEMPEÑO

Los asuntos relacionados con el desempeño son muy importantes en las redes de cómputo. Cuando hay cientos o miles de computadoras conectadas entre sí, son comunes las interacciones complejas, con consecuencias imprevisibles. Frecuentemente esta complejidad conduce a un desempeño pobre, sin que nadie sepa por qué. En las siguientes secciones examinaremos muchos temas relacionados con el desempeño de las redes para ver los tipos de problemas que existen y lo que se puede hacer para resolverlos.

Desgraciadamente, el entendimiento del desempeño de las redes es más un arte que una ciencia. Muy poca de la teoría tiene en realidad alguna utilidad en la práctica. Lo mejor que podemos hacer es dar reglas empíricas derivadas de los tropiezos y ejemplos actuales tomados del mundo real. Intencionalmente hemos postergado este análisis hasta después de estudiar la capa de transporte en las redes TCP y ATM, a fin de poder señalar los lugares en los que se han hecho bien o mal las cosas.

La capa de transporte no es el único lugar en el que surgen asuntos relacionados con el desempeño. Vimos algunos de ellos en la capa de red, en el capítulo anterior. No obstante, por lo general la capa de red está bastante ocupada con el enrutamiento y el control de congestionamientos. Los puntos más amplios, orientados al sistema, tienden a relacionarse con el transporte, por lo que este capítulo es un lugar adecuado para examinarlos.

En las siguientes cinco secciones estudiaremos cinco aspectos del desempeño de las redes:

1. Problemas de desempeño.
2. Medición del desempeño de una red.
3. Diseño de sistemas con mejor desempeño.
4. Procesamiento rápido de las TPDUs.
5. Protocolos para redes futuras de alto desempeño.

Como comentario, necesitamos un nombre para las unidades intercambiadas por las entidades de transporte. El término de TCP, segmento, es confuso en el mejor de los casos, y nunca se usa fuera del mundo TCP en este contexto. Los términos CS-PDU, SAR-PDU y CPCS-PDU son específicos de ATM. Los paquetes claramente se refieren a la capa de red y los mensajes pertenecen a la capa de aplicación. A falta de un término estándar, volveremos a llamar TPDUs a las unidades intercambiadas por las entidades de transporte. Cuando deseemos referirnos tanto a TPDUs como a paquetes, usaremos paquete como término colectivo, como en "la CPU debe ser lo bastante rápida como para procesar los paquetes de entrada en tiempo real". Con esto nos referimos tanto al paquete de capa de red como a la TPDU encapsulada en él.

6.6.1 Problemas de desempeño en las redes de cómputo

Algunos problemas de desempeño, como la congestión, son causados por sobrecargas temporales de los recursos. Si repentinamente llega más tráfico a un enrutador que el que puede manejar, surgirá la congestión y el desempeño bajará. Ya estudiamos la congestión en detalle en el capítulo anterior.

El desempeño también se degrada cuando hay un desequilibrio estructural de los recursos. Por ejemplo, si una línea de comunicación de gigabits está conectada a una PC de bajo rendimiento, la pobre CPU no será capaz de procesar los paquetes de entrada a la velocidad suficiente, y se perderán algunos. Estos paquetes se retransmitirán tarde o temprano, agregando un retardo, desperdiciando ancho de banda y reduciendo en general el desempeño.

Las sobrecargas también pueden generarse sincrónicamente. Por ejemplo, si una TPDU contiene un parámetro erróneo (por ejemplo, el puerto al que está destinada), en muchos casos el receptor cortésmente enviará una notificación de error. Ahora considere lo que podría ocurrir si se difundiera una TPDU errónea a 10,000 máquinas: cada una podría devolver un mensaje de error. La **tormenta de difusión** resultante podría paralizar la red. El UDP adoleció de este problema hasta que se cambió el protocolo para hacer que los *hosts* evitaran responder a errores en las TPDUs de UDP enviadas a direcciones de difusión.

Un segundo ejemplo de sobrecarga síncrona es lo que ocurre tras una falla del suministro eléctrico. Al regresar la energía, todas las máquinas saltan simultáneamente a sus ROMs para reiniciarse. Una secuencia de arranque común podría requerir acudir primero a algún servidor (DHCP) para conocer la identidad verdadera de la máquina, y luego a un servidor de archivos para obtener una copia del sistema operativo. Si cientos de máquinas hacen todo esto al mismo tiempo, el servidor probablemente se vendría abajo por la carga.

Aun en ausencia de sobrecargas síncronas y con suficientes recursos disponibles, puede haber un bajo desempeño debido a la falta de afinación del sistema. Por ejemplo, si una máquina tiene bastante potencia y memoria de CPU, pero no se ha asignado suficiente memoria como espacio de búfer, ocurrirán desbordamientos y se perderán varias TPDUs. De la misma manera, si el algoritmo de calendarización no tiene una prioridad bastante alta como para procesar las TPDUs de entrada, podrán perderse algunas.

Otro asunto relativo a la afinación es el establecimiento correcto de los temporizadores. Cuando se envía una TPDU, normalmente se utiliza un temporizador para protegerse contra pérdidas. Si se asigna un valor muy bajo al temporizador, ocurrirán retransmisiones innecesarias, congestionando los alambres. Si el valor es demasiado alto, ocurrirán retardos innecesarios tras la pérdida de una TPDU. Otros parámetros afinables incluyen el tiempo de espera para incorporar datos a paquetes antes de enviar confirmaciones de recepción por separado, y la cantidad de retransmisiones antes de darse por vencido.

Las redes de gigabits traen consigo nuevos problemas de desempeño. Por ejemplo, considere el envío de una ráfaga de datos de 64 KB de San Diego a Boston para llenar el búfer de 64 KB del receptor. Suponga que el enlace es de 1 Gbps y que el retardo de la luz en un sentido a través de la fibra es de 20 mseg. Inicialmente, en $t = 0$, el canal está vacío, como se muestra en la figura 6-41(a). Apenas 500 μseg después [figura 6-41(b)] todas las TPDUs están en la fibra. La TPDU a la cabeza ahora estará en algún lugar del vecindario de Brawley, todavía al sur de California. Sin embargo, el emisor debe detenerse hasta recibir la actualización de ventana.

Después de 20 mseg, la TPDU puntera llega a Boston, como se muestra en la figura 6-41(c), y se envía una confirmación de recepción. Por último, 40 mseg después de comenzar, llega la primera confirmación de recepción al emisor y puede transmitirse la segunda ráfaga. Dado que la línea de transmisión se usó durante 0.5 mseg de un total de 40, la eficiencia es de aproximadamente 1.25%. Esta situación es típica de la operación de protocolos antiguos sobre líneas de gigabits.

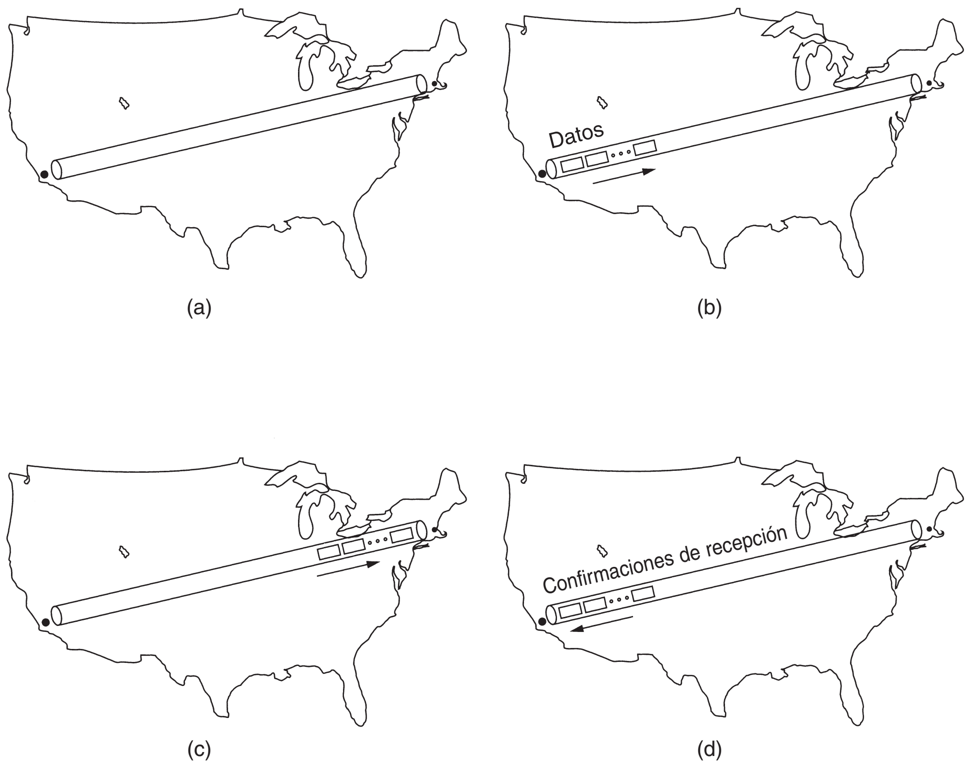


Figura 6-41. Estado de transmisión de un megabit de San Diego a Boston. (a) En $t = 0$. (b) Tras 500 μseg. (c) Tras 20 mseg. (d) Tras 40 mseg.

Una cantidad que conviene recordar durante el análisis del desempeño de redes es el **producto ancho de banda-retardo** que se obtiene al multiplicar el ancho de banda (en bits/seg) por el tiempo de retardo de ida y vuelta (en seg). El producto es la capacidad del canal desde el emisor al receptor y de regreso (en bits).

Para el ejemplo de la figura 6-41, el producto ancho de banda-retardo es de 40 millones de bits. En otras palabras, el emisor tendría que enviar una ráfaga de 40 millones de bits para trabajar a toda la velocidad hasta la llegada de la primera confirmación de recepción. Se requiere esta cantidad de bits para llenar el canal (en ambas direcciones). Ésta es la razón por la que una ráfaga de medio millón de bits sólo logra una eficiencia del 1.25%: es sólo el 1.25% de la capacidad del canal.

La conclusión aquí es que, para lograr un buen desempeño, la ventana del receptor debe tener cuando menos el tamaño del producto ancho de banda-retardo, y de preferencia ser un poco más grande, puesto que el receptor podría no responder instantáneamente. Para una línea transcontinental de gigabits se requieren cuando menos 5 megabytes para cada conexión.

Si la eficiencia es muy baja para el envío de un megabit, imagine lo que será al enviar unos cuantos cientos de bytes de una breve solicitud. A menos que pueda encontrarse otro uso para la

línea mientras el primer cliente espera una respuesta, una línea de gigabits no es mejor que una línea de megabits, sólo más cara.

Otro problema de desempeño que ocurre con las aplicaciones de tiempo crítico como audio y vídeo es la fluctuación. Un tiempo medio de transmisión corto no es suficiente. También se requiere una desviación estándar pequeña. El logro de un tiempo medio de transmisión corto con una desviación estándar pequeña requiere esfuerzos serios de ingeniería.

6.6.2 Medición del desempeño de las redes

Cuando una red tiene un desempeño pobre, sus usuarios frecuentemente se quejan con los operadores, exigiendo mejoras. Para mejorar el desempeño, los operadores deben primero determinar exactamente lo que ocurre. Para saberlo, los operadores deben efectuar mediciones. En esta sección veremos las mediciones de desempeño de las redes. El estudio siguiente se basa en el trabajo de Mogul (1993).

El ciclo usado para mejorar el desempeño de las redes contiene los siguientes pasos:

1. Medir los parámetros pertinentes y el desempeño de la red.
2. Tratar de entender lo que ocurre.
3. Cambiar un parámetro.

Estos pasos se repiten hasta que el desempeño sea lo bastante bueno o que quede claro que se han agotado todas las mejoras posibles.

Las mediciones pueden hacerse de muchas maneras y en muchos lugares (tanto físicos como en la pila de protocolos). El tipo de medición más básico es arrancar un temporizador al iniciar una actividad y medir el tiempo que tarda la actividad. Por ejemplo, saber el tiempo que toma la confirmación de recepción de una TPDU es una medición clave. Otras mediciones se hacen con contadores que registran la frecuencia con que ocurre un evento (por ejemplo, cantidad de TPDUs perdidas). Por último, con frecuencia nos interesa saber la cantidad de algo, como el número de bytes procesados durante cierto intervalo de tiempo.

La medición del desempeño y los parámetros de una red tiene muchos escollos potenciales. A continuación describimos algunos de ellos. Cualquier intento sistemático de medir el desempeño de una red debe tener cuidado de evitarlos.

Asegúrese que el tamaño de la muestra es lo bastante grande

No mida el tiempo de envío de una TPDU, sino repita la medición, digamos, un millón de veces y obtenga el promedio. Una muestra grande reducirá la incertidumbre de la media y la desviación estándar medidas. Esta incertidumbre puede calcularse usando fórmulas estadísticas estándar.

Asegúrese de que las muestras son representativas

Idealmente, la secuencia total de un millón de mediciones debería repetirse a horas del día y de la semana diferentes para ver el efecto de diferentes cargas del sistema sobre la cantidad medida. Por ejemplo, las mediciones de congestión sirven de poco si se toman en un momento en el que no hay congestión. A veces los resultados pueden ser contraintuitivos inicialmente, como la presencia de congestión intensa a las 10, 11, 13 y 14 horas, pero sin congestión al mediodía (cuando todos los usuarios están en el refrigerio).

Tenga cuidado al usar relojes de intervalos grandes

Los relojes de computadora funcionan sumando uno a un contador a intervalos regulares. Por ejemplo, un temporizador de milisegundos suma uno al contador cada 1 mseg. El uso de tal temporizador para medir un evento que tarda menos de 1 mseg es posible, pero requiere cuidado.

Por ejemplo, para medir el tiempo de envío de una TPDU, el reloj del sistema (digamos, en milisegundos) debe leerse al entrar en el código de capa de transporte, y nuevamente al salir. Si el tiempo de envío real de la TPDU es de 300 μseg, la diferencia entre las dos lecturas será 0 o 1, ambas cifras equivocadas. Sin embargo, si la medición se repite un millón de veces y se suman todas las mediciones y se dividen entre un millón, el tiempo medio tendrá una exactitud del orden de menos de 1 μseg.

Asegúrese de que no ocurre nada inesperado durante sus pruebas

Realizar mediciones en un sistema universitario el día en que tiene que entregarse un importante proyecto de laboratorio puede dar resultados diferentes a los que se podrían obtener el día siguiente. Del mismo modo, si un investigador ha decidido difundir una videoconferencia por la red mientras usted hace sus pruebas, sus resultados podrían alterarse. Es mejor ejecutar las pruebas en un sistema inactivo y crear la carga completa usted mismo, pero aun este enfoque tiene algunos escollos. Usted podría pensar que nadie usará la red a las 3 A.M., pero esa podría ser precisamente la hora en la que el programa automático de respaldos comienza a copiar todos los discos a videocinta. Es más, puede haber un tráfico pesado hacia sus fantásticas páginas del World Wide Web desde husos horarios distantes.

El caché puede arruinar las mediciones

Si quiere medir los tiempos de transferencia de archivos, la manera obvia de hacerlo es abrir un archivo grande, leerlo todo, cerrarlo y ver el tiempo que tarda. Luego se repetirá la medición muchas veces más para obtener un buen promedio. El problema es que el sistema puede manejar el archivo en caché, por lo que sólo la primera medición realmente comprende tráfico de red. Las demás son sólo lecturas del caché local. Los resultados de tales mediciones son esencialmente inservibles (a menos que se desee medir el desempeño del caché).

Con frecuencia puede evitarse el almacenamiento en caché simplemente desbordando el caché. Por ejemplo, si el caché es de 10 MB, el ciclo de prueba podría abrir, leer y cerrar dos archivos

de 10 MB en cada vuelta, en un intento por obligar a que la tasa de aciertos del caché sea de 0. Aun así, se recomienda cuidado a menos que esté completamente seguro de que entiende el algoritmo de almacenamiento en caché.

Los búferes pueden tener un efecto similar. Un programa de servicio de desempeño del TCP/IP bastante común ha llegado a informar que el UDP puede lograr un desempeño sustancialmente mayor al permitido por la línea física. ¿Por qué ocurre esto? Una llamada al UDP normalmente devuelve al control tan pronto como el *kernel* ha aceptado el mensaje y lo ha agregado a la cola de transmisión. Si hay suficiente espacio de búfer, cronometrar 1000 llamadas UDP no implica que todos los datos se han enviado. La mayoría de ellos podría estar aún en el *kernel*, pero el programa de servicio de desempeño piensa que se han transmitido todos.

Entienda lo que está midiendo

Al medir el tiempo de lectura de un archivo remoto, las mediciones dependen de la red, los sistemas operativos tanto en el cliente como en el servidor, las tarjetas de interfaz de hardware empleadas, sus controladores y otros factores. Si se tiene cuidado, finalmente se descubrirá el tiempo de transferencia de archivos para la configuración en uso. Si la meta es afinar esa configuración en particular, tales mediciones son adecuadas.

Sin embargo, si se están haciendo mediciones similares en tres sistemas diferentes a fin de seleccionar la tarjeta de interfaz de red a adquirir, sus resultados podrían desviarse por completo por el hecho de que uno de los controladores de la red realmente está mal y solamente está aprovechando el 10% del desempeño de la tarjeta.

Tenga cuidado con la extrapolación de los resultados

Suponga que hace mediciones con cargas de red simuladas que van desde 0 (en reposo) a 0.4 (40% de la capacidad), como lo indican los puntos de datos y la línea continua que los atraviesa en la figura 6-42. Puede ser tentador extrapolar linealmente, como lo indica la línea punteada. Sin embargo, muchos resultados de encolamiento comprenden un factor de $1/(1 - \rho)$, donde ρ es la carga, por lo que los valores verdaderos pueden parecerse más a la línea punteada, que se eleva más rápido que linealmente.

6.6.3 Diseño de sistemas para un mejor desempeño

La medición y los ajustes pueden con frecuencia mejorar considerablemente el desempeño, pero no pueden sustituir un buen diseño original. Una red mal diseñada puede mejorarse sólo hasta un límite. Más allá, tiene que rehacerse desde el principio.

En esta sección presentaremos algunas reglas empíricas basadas en la experiencia con muchas redes. Estas reglas se relacionan con el diseño del sistema, no sólo con el diseño de la red, ya que el software y el sistema operativo con frecuencia son más importantes que los enrutadores y las tarjetas de interfaz. La mayoría de estas ideas han sido del conocimiento común de los diseñadores

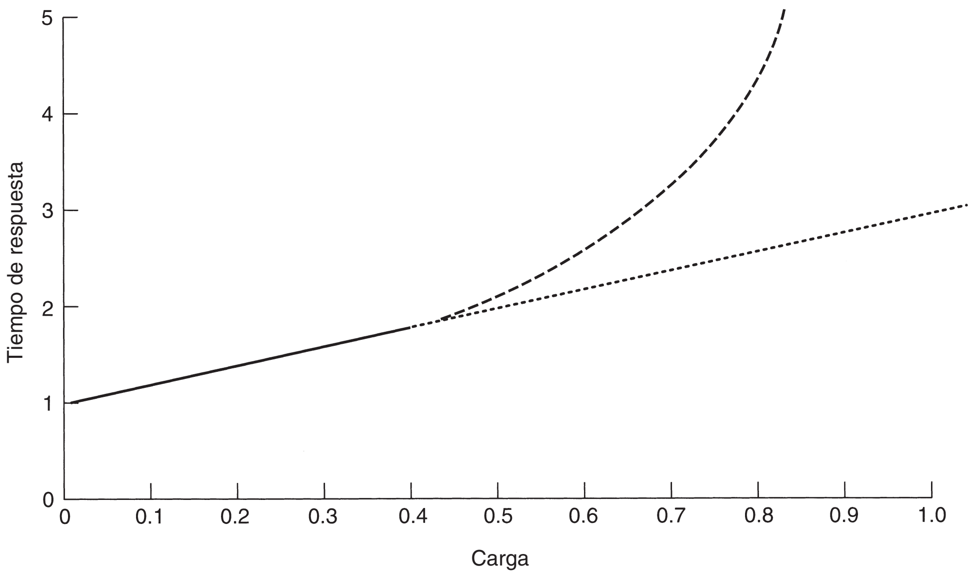


Figura 6-42. Respuesta en función de la carga.

de redes durante años y se han pasado verbalmente de generación en generación. Mogul fue el primero en postularlas explícitamente (1993); nuestro estudio sigue principalmente la secuencia del suyo. Otra fuente apropiada es (Metcalfe, 1993).

Regla #1: La velocidad de la CPU es más importante que la velocidad de la red

Una amplia experiencia ha demostrado que en casi todas las redes la sobrecarga de los sistemas operativos y protocolos domina el tiempo de utilización en el alambre. Por ejemplo, en teoría, el tiempo mínimo de una RPC en una red Ethernet es de 102 μseg, correspondiente a una solicitud mínima (64 bytes) seguida de una respuesta mínima (64 bytes). En la práctica, reducir la sobrecarga de software y conseguir el tiempo de RPC más cercano a 102 μseg es un logro considerable.

De la misma manera, el problema principal al operar a 1 Gbps es llevar los bits desde el búfer del usuario hasta la primera fibra a velocidad suficiente y lograr que la CPU receptora los procese tan rápidamente como entran. En pocas palabras, si se duplica la velocidad de la CPU, con frecuencia casi se puede duplicar la velocidad real de transporte. La duplicación de la capacidad de la red en muchos casos no tiene efecto, ya que el cuello de botella generalmente está en los *hosts*.

Regla #2: Reducir el número de paquetes para reducir la sobrecarga de software

El procesamiento de una TPDU tiene cierta cantidad de sobrecarga por TPDU (por ejemplo, procesamiento de encabezados) y cierta cantidad de procesamiento por byte (por ejemplo, procesar la suma de verificación). Al enviar 1 millón de bytes, la sobrecarga por byte es la misma sin

importar el tamaño de la TPDU. Sin embargo, el uso de TPDUs de 128 bytes implica 32 veces más sobrecarga por TPDU que el uso de TPDUs de 4 KB. Esta sobrecarga crece con rapidez.

Además de la sobrecarga de las TPDUs, hay una sobrecarga en las capas inferiores que se debe considerar. Cada paquete que llega causa una interrupción. En un procesador moderno con canalización, cada interrupción rompe la canalización de la CPU, interfiere con el caché, requiere un cambio en el contexto de administración de la memoria y obliga al almacenamiento de una cantidad de registros de CPU importante. Una reducción de n veces en las TPDUs enviadas reduce la sobrecarga de la interrupción y de los paquetes en un factor de n.

Esta observación es un argumento a favor de la recolección de una cantidad importante de datos antes de su transmisión, a fin de reducir las interrupciones en el otro lado. El algoritmo de Nagle y la solución de Clark al síndrome de la ventana tonta son intentos por lograr precisamente esto.

Regla #3: Reducir al mínimo las conmutaciones de contexto

Las conmutaciones de contexto (por ejemplo, del modo de *kernel* al modo de usuario) son mortales; pueden tener los mismos inconvenientes que las interrupciones, siendo la peor una larga serie de fallas de caché iniciales. Las conmutaciones de contexto pueden reducirse haciendo que el procedimiento de biblioteca que envía los datos los guarde en un búfer interno hasta tener una buena cantidad de ellos. De la misma manera, en el lado receptor las TPDUs de entrada pequeñas deben juntarse y pasarse al usuario como un bloque completo y no individualmente, para reducir al mínimo las conmutaciones de contexto.

En el mejor caso, un paquete entrante causa una conmutación de contexto del usuario actual al núcleo, y luego una conmutación al proceso receptor para darle los nuevos datos. Desgraciadamente, en muchos sistemas operativos ocurren conmutaciones de contexto adicionales. Por ejemplo, si el administrador de la red ejecuta un proceso especial en el espacio de usuario, un paquete entrante tenderá a causar una conmutación de contexto del usuario actual al *kernel*, luego otra del *kernel* al administrador de red, seguida de otra de regreso al *kernel* y, por último, una de éste al proceso receptor. Esta secuencia se muestra en la figura 6-43. Todas estas conmutaciones de contexto en cada paquete desperdician mucho tiempo de CPU y tienen un efecto devastador sobre el desempeño de la red.

Regla #4: Reducir al mínimo las copias

Peores que las múltiples conmutaciones de contexto son las múltiples copias. No es inusitado que un paquete entrante se copie tres o cuatro veces antes de entregarse la TPDU que contiene. Después de recibirse un paquete en la interfaz de la red en un búfer especial integrado a una tarjeta, es común que se copie en un búfer del *kernel*. De ahí se copia en el búfer de capa de red, luego en el búfer de la capa de transporte y, por último, en el proceso de aplicación receptor.

Un sistema operativo ingenioso copiará una palabra a la vez, pero no es raro que se requieran unas cinco instrucciones por palabra (carga, almacenamiento, incremento de un registro de índice,

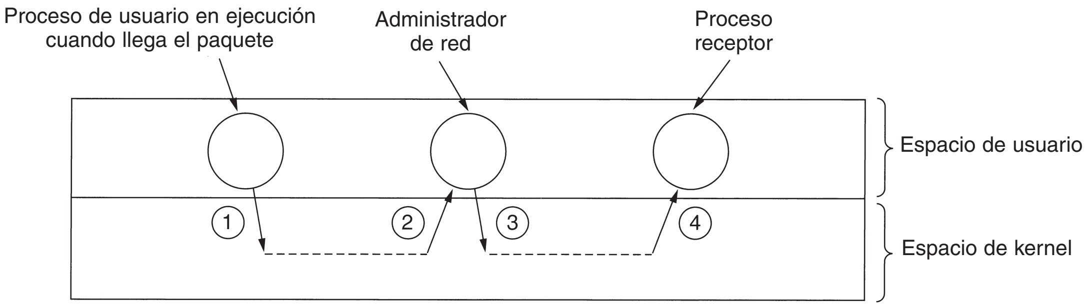


Figura 6-43. Cuatro conmutaciones de contexto para manejar un paquete con un administrador de red de espacio de usuario.

prueba de fin de datos y ramificación condicional). La elaboración de tres copias de cada paquete a cinco instrucciones por palabra de 32 bits copiada requiere 15/4 o cerca de cuatro instrucciones por byte copiado. En una CPU de 500 MIPS, una instrucción toma 2 nseg, de tal manera que cada byte necesita 8 nseg de tiempo de procesamiento o cerca de 1 nseg por bit, lo cual da una tasa máxima de 1 Gbps. Si incluimos la sobrecarga del procesamiento del encabezado, el manejo de interrupciones y las conmutaciones de contexto, podrían lograrse 500 Mbps, sin considerar el procesamiento real de los datos. Queda claro que es imposible manejar a toda velocidad una línea Ethernet de 10 Gbps.

De hecho, es probable que tampoco se pueda manejar a toda velocidad una línea de 500 Mbps. En el cálculo anterior hemos supuesto que una máquina de 500 MIPS puede ejecutar 500 millones de instrucciones por segundo. En realidad, las máquinas sólo pueden operar a tales velocidades si no hacen referencia a la memoria. Las operaciones de memoria con frecuencia son diez veces más lentas que las instrucciones registro a registro (es decir, 20 nseg/instrucción). Si en realidad 20 por ciento de las instrucciones hacen referencia a la memoria (es decir, son fallas de caché), lo cual es probable cuando entran en contacto con los paquetes entrantes, el tiempo de ejecución promedio de las instrucciones es de 5.6 nseg ($0.8 \times 2 + 0.2 \times 20$). Con cuatro instrucciones/byte, necesitamos 22.4 nseg/byte, o 2.8 nseg/bit), lo cual da cerca de 357 Mbps. Al factorizar 50 por ciento de sobrecarga da como resultado 178 Mbps. Observe que la asistencia de hardware no ayuda aquí. El problema es que el sistema operativo está ejecutando demasiadas copias.

Regla #5: Es posible comprar más ancho de banda, pero no un retardo menor

Las tres reglas que siguen tienen que ver con la comunicación, más que con el procesamiento del protocolo. La primera regla indica que, si se desea más ancho de banda, se puede comprar. La instalación de una segunda fibra junto a la primera duplica el ancho de banda, pero no hace nada para reducir el retardo. Para que el retardo sea más corto es necesario mejorar el software del protocolo, el sistema operativo o la interfaz de red. Incluso si se mejoran las tres cosas, el retardo no se reducirá si el cuello de botella es el tiempo de transmisión.

Regla #6: Evitar la congestión es mejor que recuperarse de ella

La vieja máxima de que más vale prevenir que lamentar ciertamente se aplica a la congestión en redes. Al congestionarse una red, se pierden paquetes, se desperdicia ancho de banda, se introducen retardos inútiles, y otras cosas. La recuperación requiere tiempo y paciencia; es mejor no tener que llegar a este punto. Evitar la congestión es como recibir una vacuna: duele un poco en el momento, pero evita algo que sería mucho más doloroso.

Regla #7: Evitar expiraciones del temporizador

Los temporizadores son necesarios en las redes, pero deben usarse con cuidado y deben reducirse al mínimo las expiraciones del temporizador. Al expirar un temporizador, lo común es que se repita una acción. Si realmente es necesario repetir la acción, que así sea, pero su repetición innecesaria es un desperdicio.

La manera de evitar el trabajo extra es tener cuidado de que los intervalos del temporizador sean más bien conservadores. Un temporizador que tarda demasiado en expirar agrega una pequeña cantidad de retardo extra a una conexión en el caso (improbable) de la pérdida de una TPDU. Un temporizador que expira cuando no debería consume tiempo de CPU valioso, desperdicia ancho de banda e impone una sobrecarga tal vez a docenas de enrutadores sin razón alguna.

6.6.4 Procesamiento rápido de las TPDUs

La moraleja de la historia anterior es que el obstáculo principal en las redes rápidas es el software de los protocolos. En esta sección veremos algunas maneras de acelerar este software. Para mayor información, véase (Clark y cols., 1989, y Chase y cols., 2001).

La sobrecarga de procesamiento de las TPDUs tiene dos componentes: la sobrecarga por TPDU y la sobrecarga por byte. Ambas deben combatirse. La clave para el procesamiento rápido de las TPDUs es separar el caso normal (transferencia de datos de un solo sentido) y manejarlo como caso especial. Aunque se necesita una secuencia de TPDUs especiales para entrar en el estado ESTABLISHED, una vez ahí el procesamiento de las TPDUs es directo hasta que un lado cierra la conexión.

Comencemos por examinar el lado emisor en el estado *ESTABLISHED* cuando hay datos por transmitir. Por claridad, supondremos que la entidad de transporte está en el *kernel*, aunque los mismos conceptos se aplican si es un proceso de espacio de usuario o una biblioteca en el proceso emisor. En la figura 6-44, el proceso emisor causa una interrupción en el *kernel* para ejecutar SEND. Lo primero que hace la entidad de transporte es probar si éste es el caso normal: el estado es *ESTABLISHED*, ningún lado está tratando de cerrar la conexión, se está enviando una TPDU normal (es decir, no fuera de banda) completa, y hay suficiente espacio de ventana disponible en el receptor. Si se cumplen todas las condiciones, no se requieren pruebas adicionales y puede seguirse la trayectoria rápida a través de la entidad de transporte emisora. Por lo general, esta ruta se toma la mayoría de las veces.

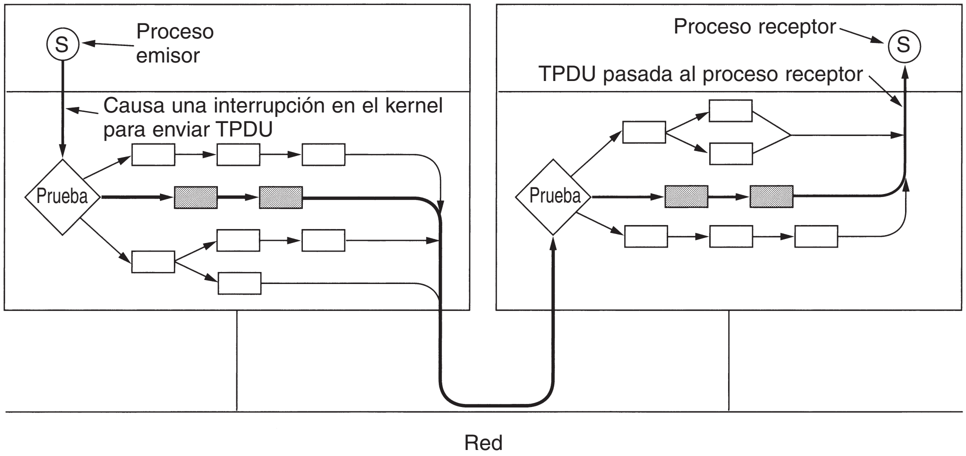


Figura 6-44. La trayectoria rápida del emisor al receptor se indica con una línea gruesa. Los pasos de procesamiento de esta trayectoria se muestran sombreados.

En el caso normal, los encabezados de las TPDUs de datos consecutivas son casi iguales. Para aprovechar este hecho, se almacena un encabezado prototipo en la entidad de transporte. Al principio de la trayectoria rápida, el encabezado se copia lo más rápidamente posible en un búfer de trabajo, palabra por palabra. Los campos que cambian de una TPDU a otra se sobreescriben en el búfer. Con frecuencia, estos campos se deducen fácilmente de las variables de estado, como el siguiente número de secuencia. A continuación se pasan a la capa de red un apuntador al encabezado completo de la TPDU más un apuntador a los datos de usuario. Aquí puede seguirse la misma estrategia (no se muestra en la figura 6-44). Por último, la capa de red entrega el paquete resultante a la capa de enlace de datos para su transmisión.

Como ejemplo del funcionamiento de este principio en la práctica, consideremos el TCP/IP. En la figura 6-45(a) se muestra el encabezado TCP. Los campos que son iguales en varias TPDUs consecutivas durante un flujo en un solo sentido aparecen sombreados. Todo lo que tiene que hacer la entidad de transporte emisora es copiar las cinco palabras del encabezado prototipo en el búfer de salida, actualizar el número de secuencia (copiándolo de una palabra en la memoria), calcular la suma de verificación e incrementar el número de secuencia en la memoria. Entonces puede entregar el encabezado y los datos a un procedimiento IP especial para enviar una TPDU normal máxima. El IP entonces copia su encabezado prototipo de cinco palabras [véase la figura 6-45(b)] en el búfer, llena el campo de *Identificación* y calcula su suma de verificación. El paquete ya está listo para transmitirse.

Veamos ahora el proceso de trayectoria rápida del lado receptor de la figura 6-44. El paso 1 es localizar el registro de conexión de la TPDU entrante. En el TCP, el registro de conexión puede almacenarse en una tabla de *hash* en la que alguna función sencilla de las dos direcciones IP y los dos puertos es la clave. Una vez localizado el registro de conexión, ambas direcciones y ambos puertos deben compararse para comprobar que se ha encontrado el registro correcto.

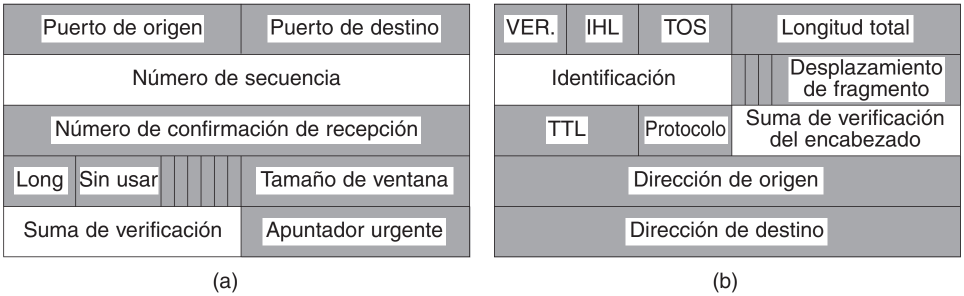


Figura 6-45. (a) Encabezado TCP. (b) Encabezado IP. En ambos casos, los campos sombreados se toman sin cambios del prototipo.

Una optimización que con frecuencia acelera aún más la búsqueda del registro de conexión es sencilla: mantener un apuntador al último registro usado y probar ese primero. Clark y cols. (1989) probó esto y observó una tasa de éxito mayor al 90%. Otras heurísticas de búsqueda se describen en (McKenney y Dove, 1992).

A continuación se revisa la TPDU para ver si es normal: el estado es *ESTABLISHED*, ninguno de los dos lados está tratando de cerrar la conexión, la TPDU es completa, no hay indicadores especiales encendidos y el número de secuencia es el esperado. Estas pruebas se llevan apenas unas cuantas instrucciones. Si se cumplen todas las condiciones, se invoca un procedimiento TCP especial de trayectoria rápida.

La trayectoria rápida actualiza el registro de la conexión y copia los datos en el espacio de usuario. Mientras copia, el procedimiento también calcula la suma de verificación, eliminado una pasada extra por los datos. Si la suma de verificación es correcta, se actualiza el registro de conexión y se devuelve una confirmación de recepción. El esquema general de hacer primero una comprobación rápida para ver si el encabezado es el esperado y tener un procedimiento especial para manejar ese caso se llama **predicción de encabezado.** Muchas implementaciones del TCP lo usan. Cuando esta optimización y todas las demás estudiadas en este capítulo se usan juntas, es posible conseguir que el TCP opere al 90% de la velocidad de una copia local de memoria a memoria, suponiendo que la red misma es lo bastante rápida.

Dos áreas en las que son posibles mejoras sustanciales del desempeño son el manejo de búferes y la administración de los temporizadores. El aspecto importante del manejo de búferes es evitar el copiado innecesario, como se explicó antes. La administración de los temporizadores es importante porque casi ninguno de los temporizadores expira; se ajustan para protegerse contra pérdidas de TPDUs, pero la mayoría de las TPDUs llegan correctamente, y sus confirmaciones de recepción también. Por tanto, es importante optimizar el manejo de los temporizadores para que casi nunca expiren.

Un esquema común consiste en usar una lista enlazada de eventos del temporizador ordenada por hora de expiración. La entrada inicial contiene un contador que indica la cantidad de pulsos de reloj que faltan para la expiración. Cada entrada subsecuente contiene un contador que indica el rezago en pulsos después de la entrada previa. Por tanto, si los temporizadores expiran a 3, 10 y 12 pulsos, respectivamente, los tres contadores son de 3, 7 y 2, respectivamente.

Después de cada pulso de reloj, se decrementa el contador del encabezado inicial. Cuando llega a cero, su evento se procesa y el siguiente elemento de la lista es ahora el inicial; su contador no necesita cambiarse. En este esquema, la inserción y eliminación de temporizadores son operaciones costosas, con tiempos de ejecución proporcionales a la longitud de la lista.

Puede usarse un enfoque más eficiente si el intervalo máximo del temporizador está limitado y se conoce por adelantado. Aquí puede utilizarse un arreglo llamado **rueda de temporización,** como se muestra en la figura 6-46. Cada ranura corresponde a un pulso de reloj. El tiempo actual en la figura es $T = 4$. Los temporizadores se programan para expirar 3, 10 y 12 pulsos más adelante. Si se establece un temporizador nuevo para expirar en siete pulsos, simplemente se crea una entrada en la ranura 11. Del mismo modo, si el temporizador establecido para $T + 10$ tiene que cancelarse, debe examinarse la lista que comienza en la ranura 14 para eliminar la entrada pertinente. Observe que el arreglo de la figura 6-46 no puede manejar temporizadores más allá de $T + 15$.

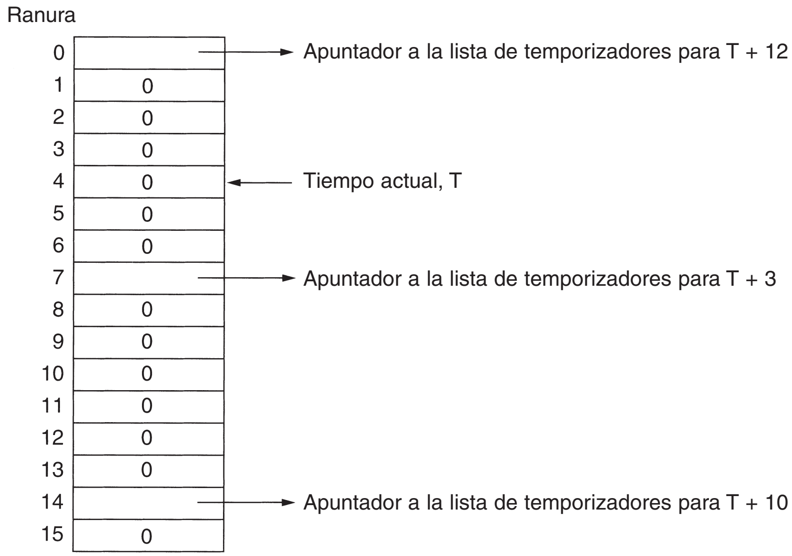


Figura 6-46. Rueda de temporización.

Cuando el reloj pulsa, el apuntador de tiempo actual avanza una ranura (circularmente). Si la entrada a la que ahora se apunta es diferente de cero, todos sus temporizadores se procesan. Se estudian muchas variaciones de la idea básica en (Varghese y Lauck, 1987).

6.6.5 Protocolos para redes de gigabits

Al inicio de la década de 1990 comenzaron a aparecer las redes de gigabits. La primera reacción de la gente fue usar en ellas los viejos protocolos, pero pronto surgieron varios problemas. En esta sección estudiaremos algunos de estos problemas y las medidas que están tomando los protocolos nuevos para resolverlos conforme surgen redes todavía más rápidas.

El primer problema es que muchos protocolos usan números de secuencia de 32 bits. En los principios de Internet, las líneas entre enrutadores fueron en su mayoría líneas rentadas de 56 kbps, así que un *host* transmitiendo a toda velocidad tomaba alrededor de una semana para dar vuelta a los números de secuencia. Para los diseñadores de TCP, 2^{32} era una aproximación muy buena al infinito porque había poco riesgo de que los paquetes viejos deambularan por la red una semana después de su transmisión. Con la Ethernet de 10 Mpbs, el tiempo para dar vuelta a los números de secuencia se redujo a 57 minutos, mucho más corto pero aún manejable. Con una Ethernet de 1 Gbps sirviendo datos en Internet, el tiempo para dar vuelta a los números de secuencia es cercano a 34 segundos, bastante abajo de los 120 seg de tiempo de vida máximo de un paquete en Internet. De buenas a primeras, 2^{32} ya no es una buena aproximación al infinito porque un emisor puede recorrer el espacio de secuencia aunque los paquetes viejos aún existan en la red. No obstante, el RFC 1323 proporciona una ventana de escape.

El origen de este problema es que muchos diseñadores de protocolos simplemente supusieron, tácitamente, que el tiempo requerido para consumir el espacio de secuencia completo excedería por mucho el tiempo de vida máximo de los paquetes. En consecuencia, no había necesidad de preocuparse por el problema de que duplicados viejos sobrevivieran aún al dar vuelta a los números de secuencia. A velocidades de gigabits, falla ese supuesto implícito.

Un segundo problema es que las velocidades de comunicación han mejorado con mucha mayor rapidez que las velocidades de cómputo. (Nota a los ingenieros en computación: ¡salgan a darles una paliza a los ingenieros en comunicaciones! Contamos con ustedes.) En los años 70, ARPANET operaba a 56 kbps y tenía computadoras que operaban a aproximadamente 1 MIPS. Los paquetes eran de 1008 bits, por lo que ARPANET era capaz de entregar unos 56 paquetes/seg. Con casi 18 mseg disponibles por paquete, un *host* podía darse el lujo de gastar 18,000 instrucciones en el procesamiento de un paquete. Claro que hacerlo requería todo el tiempo de CPU, pero podían ejecutarse 9000 instrucciones por paquete y aún así tener la mitad del tiempo de CPU libre para llevar a cabo tareas reales.

Compare estas cantidades con las computadoras modernas de 1000 MIPS que intercambian paquetes de 1500 bytes por una línea de gigabits. Los paquetes pueden entrar a una velocidad de más de 80,000 por segundo, por lo que el procesamiento de los paquetes debe completarse en 6.25 μseg si queremos reservar la mitad de la CPU para las aplicaciones. En 6.25 μseg, una computadora de 1000 MIPS puede ejecutar 6250 instrucciones, apenas 1/3 de lo que tenían disponibles los *hosts* de ARPANET. Es más, las instrucciones RISC modernas hacen menos por instrucción de lo que hacían las viejas instrucciones CISC, por lo que el problema es aún peor de lo que parece. En conclusión, hay menos tiempo disponible para el procesamiento de los protocolos del que había antes, por lo que los protocolos deben volverse más sencillos.

Un tercer problema es que el protocolo de retroceso n se desempeña mal en las líneas con un producto ancho de banda-retardo grande. Por ejemplo, considere una línea de 4000 km que opera a 1 Gbps. El tiempo de transmisión de ida y vuelta es de 40 mseg, durante el cual un emisor puede enviar 5 megabytes. Si se detecta un error, pasarán 40 mseg antes de que el emisor se entere. Si se usa el retroceso n, el emisor tendrá que transmitir no sólo el paquete erróneo, sino también los 5 megabytes de paquetes que llegaron después. Evidentemente, éste es un desperdicio masivo de recursos.

Un cuarto problema es que las líneas de gigabits son fundamentalmente diferentes de las líneas de megabits en el sentido de que las líneas largas de gigabits están limitadas por el retardo en

lugar del ancho de banda. En la figura 6-47 mostramos el tiempo que tarda la transferencia de un archivo de 1 megabit a 4000 km con varias velocidades de transmisión. A velocidades de hasta 1 Mbps, el tiempo de transmisión está dominado por la velocidad con que pueden enviarse los bits. A 1 Gbps, el retardo de ida y vuelta de 40 mseg domina al 1 mseg que se tarda la inserción de los bits en la fibra. Aumentos mayores del ancho de banda apenas tienen algún efecto.

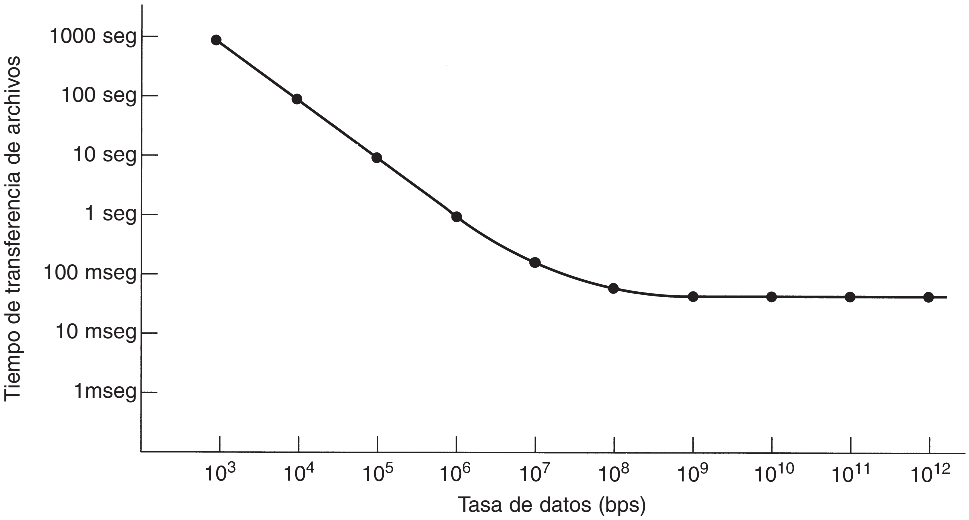


Figura 6-47. Tiempo de transferencia y confirmación de recepción de un archivo de 1 megabit a través de una línea de 4000 km.

La figura 6-47 tiene implicaciones desafortunadas para los protocolos de red; indica que los protocolos de parada y espera, como el RPC, tienen un límite superior de desempeño inherente. Este límite lo establece la velocidad de la luz. Ningún progreso tecnológico en la óptica mejorará la situación (aunque ayudarían nuevas leyes de la física).

Un quinto problema que vale la pena mencionar no es tecnológico ni de protocolo, como los otros, sino resultado de las aplicaciones nuevas. En pocas palabras, en muchas aplicaciones de gigabits, como las de multimedia, la variación en los tiempos de llegada de los paquetes es tan importante como el retardo medio mismo. Una tasa de entrega lenta pero uniforme con frecuencia es preferible a una rápida pero variable.

Pasemos ahora de los problemas a las maneras de resolverlos. Primero haremos algunas observaciones generales, luego veremos los mecanismos de protocolos, la disposición de los paquetes y el software de los protocolos.

El principio básico que deben aprender de memoria todos los diseñadores de redes de gigabits es:

Diseñar pensando en la velocidad, no en la optimización del ancho de banda.

Los protocolos viejos con frecuencia se diseñaban tratando de reducir al mínimo la cantidad de bits en el alambre, comúnmente usando campos pequeños y empacándolos en bytes y palabras.

Hoy día hay más que suficiente ancho de banda. El procesamiento del protocolo es el problema, por lo que los protocolos deberían diseñarse para reducirlo al mínimo. Los diseñadores del IPv6 entendieron claramente este principio.

Una manera tentadora de acelerar el procedimiento es construir interfaces de red rápidas en hardware. Lo malo de esta estrategia es que, a menos que el protocolo sea excesivamente sencillo, "hardware" simplemente significa una tarjeta con una segunda CPU y su propio programa. Para asegurar que el coprocesador de la red sea más económico que la CPU principal, con frecuencia se usa un chip más lento. La consecuencia de este diseño es que una buena parte del tiempo la CPU principal (rápida) queda en espera de que la segunda CPU (lenta) haga el trabajo crítico. Es un mito pensar que la CPU principal tiene otras tareas que hacer mientras espera. Es más, cuando dos CPUs de propósito general se comunican, pueden ocurrir condiciones de competencia, por lo que se requieren protocolos complejos entre los dos procesadores para sincronizarlos correctamente. Por lo general, el mejor enfoque es hacer que los protocolos sean sencillos y dejar que la CPU principal realice el trabajo.

Veamos ahora el asunto de la retroalimentación en los protocolos de alta velocidad. Debido al ciclo de retardo (relativamente) largo, debe evitarse la retroalimentación: la señalización del receptor al emisor tarda demasiado. Un ejemplo de retroalimentación es el control de la tasa de transmisión mediante un protocolo de ventana corrediza. Para evitar los retardos (largos) inherentes en el envío de actualizaciones de ventana del receptor al emisor, es mejor usar un protocolo basado en la tasa. En tal protocolo, el emisor puede enviar todo lo que quiera, siempre y cuando no envíe a mayor velocidad que cierta tasa acordada de antemano entre el emisor y el receptor.

Un segundo ejemplo de retroalimentación es el algoritmo de arranque lento de Jacobson. Este algoritmo efectúa múltiples sondeos para saber qué tanto puede manejar la red. Con las redes de alta velocidad, efectuar media docena de sondeos pequeños para ver la respuesta de la red es un desperdicio de ancho de banda enorme. Un esquema más eficiente es hacer que el emisor, el receptor y la red reserven los recursos necesarios en el momento de establecer la conexión. La reservación de recursos por adelantado también tiene la ventaja de facilitar la reducción de la fluctuación. En pocas palabras, la elevación de las velocidades inevitablemente empuja el diseño hacia la operación orientada a la conexión, o a algo muy parecido. Por supuesto, si el ancho de banda se incrementa en el futuro en forma tal que a nadie le importe desperdiciarlo, las reglas de diseño se tornarán muy diferentes.

La disposición de los paquetes es una consideración importante en las redes de gigabits. El encabezado debería contener la menor cantidad de campos posible, a fin de reducir el tiempo de procesamiento; estos campos deben ser lo bastante grandes como para cumplir su cometido, y estar alineados con los límites de palabra para facilitar su procesamiento. En este contexto, "bastante grandes" significa que no ocurren problemas como números de secuencia que dan vuelta mientras existen aún paquetes viejos, receptores incapaces de anunciar suficiente espacio de ventana porque el campo de ventana es demasiado pequeño, etcétera.

El encabezado y los datos deben tener sumas de verificación aparte, por dos razones. Primera, hacer posible la obtención de la suma de verificación del encabezado, pero no de los datos. Segunda, comprobar que el encabezado esté correcto antes de comenzar a copiar los datos en el espacio de usuario. Es deseable calcular la suma de verificación de los datos al mismo tiempo que

se copian los datos en el espacio de usuario, pero si el contenido del encabezado está equivocado, el copiado podría hacerse a un proceso equivocado. Para evitar un copiado incorrecto pero permitir que se calcule la suma de verificación de los datos durante el copiado, es esencial que las dos sumas de verificación estén separadas.

El tamaño máximo de los datos debe ser lo bastante grande para permitir una operación eficiente inclusive ante retardos largos. También, cuanto mayor es el tamaño del bloque de datos, menor es la fracción del ancho de banda total dedicada a los encabezados. 1500 bytes es demasiado pequeño.

Otra característica valiosa es la capacidad de enviar una cantidad normal de datos junto con la solicitud de conexión. Así, puede ahorrarse el tiempo de un viaje de ida y vuelta.

Por último, es importante decir algo sobre el software de los protocolos. Una idea clave es concentrarse en el caso exitoso. Muchos protocolos viejos tienden a remarcar lo que se debe hacer cuando algo falla (por ejemplo, cuando se pierde un paquete). Para lograr que los protocolos operen rápidamente, el diseñador debería enfocarse en reducir al mínimo el tiempo de procesamiento cuando todo funciona bien. La reducción al mínimo del tiempo de procesamiento cuando ocurren errores es secundaria.

Un segundo asunto relacionado con el software es la minimización del tiempo de copiado. Como vimos antes, el copiado de datos con frecuencia es la fuente principal de sobrecarga. Idealmente, el hardware debe poner en la memoria cada paquete entrante como un bloque contiguo de datos. El software entonces debe copiar este paquete en el búfer del usuario con un solo copiado de bloque. Dependiendo del modo de funcionamiento del caché, inclusive puede ser deseable evitar un ciclo de copiado. En otras palabras, para copiar 1024 palabras, la manera más rápida podría ser la ejecución de 1024 instrucciones MOVE una tras otra (o 1024 pares cargar-almacenar). La rutina de copiado es tan crítica que debe desarrollarse cuidadosamente en código ensamblador, a menos que haya una manera de lograr que el compilador produzca el código óptimo.

6.7 RESUMEN

La capa de transporte es la clave para entender los protocolos en capas. Esta capa proporciona varios servicios, siendo el más importante un flujo de bytes punto a punto, confiable y orientado a al conexión desde el emisor hasta el receptor. Se accede a ella a través de primitivas de servicio que permiten establecer, usar y liberar conexiones. Una interfaz común de la capa de transporte es la que proporcionan los *sockets* de Berkeley.

Los protocolos de transporte deben ser capaces de administrar conexiones a través de redes no confiables. El establecimiento de conexiones se complica por la existencia de paquetes duplicados retardados que pueden reaparecer en momentos inoportunos. Para manejarlos, se requieren acuerdos de tres vías para establecer las conexiones. La liberación de una conexión es más sencilla que su establecimiento, pero aun así no está exento de complejidad debido al problema de los dos ejércitos.

Aun si la capa de red es completamente confiable, la capa de transporte tiene bastante trabajo: debe manejar todas las primitivas de servicio, administrar conexiones y temporizadores y asignar y usar créditos.

Internet tiene dos protocolos de transporte principales: UDP y TCP. UDP es un protocolo no orientado a la conexión que es principalmente una envoltura para los paquetes IP con la característica adicional de multiplexar y desmultiplexar diversos procesos utilizando una sola dirección IP. UDP puede emplearse para interacciones cliente-servidor, por ejemplo, utilizando RPC. También puede emplearse para construir protocolos en tiempo real como RTP.

El protocolo de transporte principal de Internet es TCP. Proporciona un flujo de bytes bidireccional y confiable. Utiliza un encabezado de 20 bytes en todos los segmentos. Los enrutadores de Internet pueden fragmentar los segmentos, por lo que los *hosts* deben estar preparados para reensamblarlos. Se ha invertido mucho trabajo en optimizar el desempeño del TCP, mediante algoritmos de Nagle, Clark, Jacobson, Karn, entre otros. Los enlaces inalámbricos agregan una variedad de complicaciones al TCP. El TCP para transacciones es una extensión del TCP que maneja las interacciones cliente-servidor con un número reducido de paquetes.

El desempeño de las redes generalmente es dominado por la sobrecarga de procesamiento de los protocolos y las TPDUs, y esta situación empeora a mayores velocidades. Los protocolos deberían diseñarse para reducir al mínimo la cantidad de TPDUs, de conmutaciones de contexto y de veces que se copia cada TPDU. En las redes de gigabits se requieren protocolos sencillos.

PROBLEMAS

1. En nuestras primitivas de ejemplo de la figura 6-2, LISTEN es una llamada bloqueadora. ¿Es estrictamente necesario esto? De no serlo, explique cómo debe usarse una primitiva no bloqueadora. ¿Qué ventaja tendría esto respecto al esquema descrito en el texto?

2. En el modelo de la figura 6-4, se supone que la capa de red puede perder paquetes y, por tanto, su recepción se debe confirmar individualmente. Suponga que la capa de red es 100% confiable y que nunca pierde paquetes. ¿Qué cambios, si acaso, se necesitarán en la figura 6-4?

3. En las dos partes de la figura 6-6 hay un comentario de que los valores de *SERVER_PORT* deben ser los mismos en el cliente y en el servidor. ¿Por qué es tan importante?

4. Suponga que el esquema operado por reloj para generar números de secuencia iniciales se usa con un contador de reloj de 15 bits de ancho. El reloj pulsa una vez cada 100 mseg, y el tiempo de vida máximo de un paquete es de 60 seg. ¿Con qué frecuencia ocurre la resincronización
(a) en el peor caso?
(b) cuando los datos consumen 240 números de secuencia/min?

5. ¿Por qué tiene que ser el tiempo de vida máximo de paquete, *T*, lo bastante grande para asegurar que han desaparecido no sólo el paquete, sino también sus confirmaciones de recepción?

6. Imagine que se usa un acuerdo de dos vías en lugar de uno de tres vías para establecer las conexiones. En otras palabras, no se requiere el tercer mensaje. ¿Son posibles ahora los bloqueos irreversibles? Dé un ejemplo o demuestre que no pueden existir.

7. Imagine un problema de *n*-ejércitos generalizado, en el que el acuerdo de dos de los ejércitos azules es suficiente para la victoria. ¿Existe un protocolo que permita ganar a los azules?

8. Considere el problema de la recuperación después de una caída del *host* (es decir, la figura 6-18). Si el intervalo entre la escritura y el envío de una confirmación de recepción, o viceversa, puede hacerse relativamente pequeño, ¿cuáles son las mejores estrategias emisor-receptor para reducir al mínimo la posibilidad de una falla del protocolo?

9. ¿Son posibles los bloqueos irreversibles con la entidad de transporte descrita en el texto (figura 6-20)?

10. Por curiosidad, el implementador de la entidad de transporte de la figura 6-20 ha decidido incluir contadores en el procedimiento *sleep* para obtener estadísticas sobre el arreglo *conn*. Entre éstas están la cantidad de conexiones en cada uno de los siete estados posibles, n_i (i = 1,..., 7). Tras escribir un enorme programa en FORTRAN para analizar los datos, nuestro implementador descubrió que la relación $\Sigma n_i = MAX_CONN$ parece ser verdadera siempre. ¿Hay otras invariantes en las que intervengan sólo estas siete variables?

11. ¿Qué ocurre cuando el usuario de la entidad de transporte dada en la figura 6-20 envía un mensaje de longitud cero? Explique el significado de su respuesta.

12. Por cada evento que puede ocurrir en la entidad de transporte de la figura 6-20, indique si es legal o no cuando el usuario está durmiendo en el estado *sending.*

13. Explique las ventajas y desventajas de los créditos en comparación con los protocolos de ventana corrediza.

14. ¿Por qué existe el UDP? ¿No habría bastado con permitir que los procesos de usuario enviaran paquetes IP en bruto?

15. Considere un protocolo de nivel de aplicación simple construido encima de UDP que permite a un cliente recuperar un archivo desde un servidor remoto que reside en una dirección bien conocida. El cliente primero envía una solicitud con el nombre del archivo, y el servidor responde con una secuencia de paquetes de datos que contienen diferentes partes del archivo solicitado. Para asegurar la confiabilidad y una entrega en secuencia, el cliente y el servidor utilizan un protocolo de parada y espera. Ignorando el aspecto de desempeño obvio, ¿ve usted un problema con este protocolo? Piense cuidadosamente en la posibilidad de la caída de los procesos.

16. Un cliente envía una solicitud de 128 bytes a un servidor localizado a 100 km de distancia a través de una fibra óptica de 1 gigabit. ¿Cuál es la eficiencia de la línea durante la llamada a procedimiento remoto?

17. Considere nuevamente la situación del problema anterior. Calcule el tiempo de respuesta mínimo posible para la línea de 1 Gbps y para una de 1 Mbps. ¿Qué conclusión puede obtener?

18. Tanto UDP como TCP utilizan números de puerto para identificar la entidad de destino cuando entregan un paquete. Dé dos razones por las cuales estos protocolos inventaron un nuevo ID abstracto (números de puerto), en lugar de utilizar IDs de proceso, que ya existían cuando se diseñaron estos protocolos.

19. ¿Cuál es el tamaño total de la MTU mínima de TCP, incluyendo la sobrecarga de TCP e IP pero no la de la capa de enlace de datos?

20. La fragmentación y el reensamble de datagramas son manejados por IP y son transparentes para TCP. ¿Esto significa que TCP no tiene que preocuparse porque los datos lleguen en el orden equivocado?

21. RTP se utiliza para transmitir audio con calidad de CD, el cual crea un par de muestras de 16 bits, 44,100 veces/seg, una muestra por cada uno de los canales de estéreo. ¿Cuántos paquetes por segundo debe transmitir RTP?

22. ¿Sería posible colocar el código RTP en el *kernel* del sistema operativo junto con el código UDP? Explique su respuesta.

23. Un proceso del *host* 1 se ha asignado al puerto *p*, y un proceso del *host* 2 se ha asignado al puerto *q*. ¿Es posible que haya dos o más conexiones TCP entre estos dos puertos al mismo tiempo?

24. En la figura 6-29 vimos que además del campo *Confirmación de recepción* de 32 bits, hay un bit *ACK* en la cuarta palabra. ¿Esto agrega realmente algo? ¿Por qué sí o por qué no?

25. La máxima carga útil de un segmento TCP son 65,495 bytes. ¿Por qué se eligió ese extraño número?

26. Describa dos formas de entrar en el estado *SYN RCVD* de la figura 6-33.

27. Indique una desventaja potencial del uso del algoritmo de Nagle en una red muy congestionada.

28. Considere el efecto de usar arranque lento en una línea con un tiempo de ida y vuelta de 10 mseg sin congestionamientos. La ventana receptora es de 24 KB y el tamaño máximo de segmento es de 2 KB. ¿Cuánto tiempo pasará antes de poder enviar la primera ventana completa?

29. Suponga que la ventana de congestionamiento del TCP está ajustada a 18 KB y que ocurre una expiración del temporizador. ¿Qué tan grande será la ventana si las siguientes cuatro ráfagas de transmisiones tienen éxito? Suponga que el tamaño máximo de segmento es de 1 KB.

30. Si el tiempo de ida y vuelta del TCP, *RTT*, actualmente es de 30 mseg y las siguientes confirmaciones de recepción llegan después de 26, 32 y 24 mseg, respectivamente, ¿cuál es la nueva estimación de *RTT* utilizando el algoritmo de Jacobson? Use $\alpha = 0.9$.

31. Una máquina TCP envía ventanas de 65,535 bytes por un canal de 1 Gbps que tiene un retardo de 10 mseg en un solo sentido. ¿Cuál es la velocidad real de transporte máxima que se puede lograr? ¿Cuál es la eficiencia de la línea?

32. ¿Cuál es la velocidad máxima de línea a la que un *host* puede enviar cargas útiles TCP de 1500 bytes con un tiempo de vida máximo de paquete de 120 seg sin que los números de secuencia den vuelta? Tome en cuenta la sobrecarga TCP, IP y Ethernet. Suponga que las tramas Ethernet se pueden enviar de manera continua.

33. En una red que tiene un tamaño máximo de TPDU de 128 bytes, un tiempo de vida máximo de TPDU de 30 seg, y un número de secuencia de 8 bits, ¿cuál es la tasa máxima de datos por conexión?

34. Suponga que usted está midiendo el tiempo para recibir una TPDU. Cuando ocurre una interrupción, lee el reloj del sistema en milisegundos. Cuando la TPDU se ha procesado por completo, lee nuevamente el reloj. Mide 0 mseg 270,000 veces y 1 mseg 730,000 veces. ¿Cuánto tarda en recibir una TPDU?

35. Una CPU ejecuta instrucciones a la tasa de 1000 MIPS. Los datos pueden copiarse 64 bits a la vez, y cada palabra toma diez instrucciones para copiarse. Si un paquete entrante tiene que copiarse cuatro veces, ¿puede este sistema manejar una línea de 1 Gbps? Por simplicidad, suponga que todas las instrucciones, incluso aquellas que leen o escriben en la memoria, se ejecutan a la tasa total de 1000 MIPS.

36. Para resolver el problema de los números de secuencia repetidos mientras que los paquetes anteriores aún existen, se podrían utilizar números de secuencia de 64 bits. Sin embargo, teóricamente, una fibra óptica puede ejecutarse a 75 Tbps. ¿Cuál es el tiempo de vida máximo de paquete que se requiere para asegurarse de que las futuras redes de 75 Tbps no tengan problemas de números de secuencia que den vuelta incluso con los números de secuencia de 64 bits? Suponga que cada byte tiene su propio número de secuencia, al igual que TCP.

37. Mencione una ventaja de RPC sobre UDP en comparación con el TCP para transacciones. Mencione una ventaja del T/TCP sobre RPC.

38. En la figura 6-40(a) vimos que para completar el RCP se necesitan 9 paquetes. ¿Hay alguna circunstancia en la que se necesiten exactamente 10 paquetes?

39. En la sección 6.6.5 calculamos que una línea de gigabits descarga 80,000 paquetes/seg en el *host*, y éste dispone de sólo 6250 instrucciones para procesarlos, lo que deja la mitad del tiempo de CPU para las aplicaciones. Para este cálculo se tomó un tamaño de paquete de 1500 bytes. Rehaga el cálculo para un paquete de tamaño ARPANET (128 bytes). En ambos casos, suponga que los tamaños de paquete dados incluyen toda la sobrecarga.

40. Para una red de 1 Gbps que opera sobre 4000 km, el retardo es el factor limitante, no el ancho de banda. Considere una MAN con un promedio de 20 km entre el origen y el destino. ¿A qué tasa de datos el retardo de ida y vuelta debido a la velocidad de la luz iguala el retardo de transmisión para un paquete de 1 KB?

41. Calcule el producto de retardo de ancho de banda para las siguientes redes: (1) T1 (1.5 Mbps), (2) Ethernet (10 Mbps), (3) T3 (45 Mbps) y (4) STS-3 (155 Mbps). Suponga un RTT de 100 mseg. Recuerde que un encabezado TCP tiene 16 bits reservados para el tamaño de ventana. ¿Cuáles son sus implicaciones a la luz de sus cálculos?

42. ¿Cuál es el producto de retardo de ancho de banda para un canal de 50 Mbps en un satélite geoestacionario? Si todos los paquetes son de 1500 bytes (incluyendo la sobrecarga), ¿qué tan grande debería ser la ventana en los paquetes?

43. El servidor de archivos de la figura 6-6 está muy lejos de ser perfecto y se le pueden hacer algunas mejoras. Realice las siguientes modificaciones.
(a) Dé al cliente un tercer argumento que especifique un rango de bytes.
(b) Agregue un indicador–w al cliente que permita que el archivo se escriba en el servidor.

44. Modifique el programa de la figura 6-20 para que realice recuperación de errores. Agregue un nuevo tipo de paquetes, *reset*, que pueda llegar después de que los dos lados abran una conexión pero que ninguno la cierre. Este evento, que sucede de manera simultánea en ambos lados de la conexión, significa que cualquier paquete que estuviera en tránsito se ha entregado o destruido, pero de cualquier modo ya no está en la subred.

45. Escriba un programa que simule manejo de búfer en una entidad de transporte, utilizando una ventana corrediza para el control de flujo en lugar del sistema de créditos de la figura 6-20. Deje que los procesos de las capas superiores abran conexiones, envíen datos y cierren las conexiones de manera aleatoria. Por sencillez, haga que los datos viajen sólo de la máquina *A* a la máquina *B*. Experimente con estrategias de asignación de búfer diferentes en *B*, como dedicar búferes a conexiones específicas y establecer un grupo de búferes común, y mida el rendimiento total alcanzado por cada estrategia.

46. Diseñe e implemente un sistema de salones de conversación que permita conversar a múltiples grupos de usuarios. Un coordinador del salón reside en una dirección bien conocida de red, utiliza UDP para comunicarse con los clientes del salón, configura los servidores del salón para cada sesión de conversación y mantiene un directorio de sesiones de conversación. Hay un servidor de conversación por sesión. Un servidor de este tipo utiliza TCP para comunicarse con los clientes. Un cliente de conversación permite que los usuarios inicien, se unan y dejen una sesión de conversación. Diseñe e implemente el código del coordinador, del servidor y del cliente.

7

LA CAPA DE APLICACIÓN

Habiendo concluido todos los preliminares, ahora llegamos a la capa de aplicación, donde pueden encontrarse todas las aplicaciones interesantes. Las capas por debajo de la de aplicación están ahí para proporcionar transporte confiable, pero no hacen ningún trabajo verdadero para los usuarios. En este capítulo estudiaremos algunas aplicaciones de red reales.

Sin embargo, aun en la capa de aplicación se necesitan protocolos de apoyo que permitan el funcionamiento de las aplicaciones reales. De manera acorde, veremos uno de éstos antes de que comencemos con las aplicaciones mismas. El sujeto en cuestión es el DNS, que maneja la asignación de nombres dentro de Internet. Después examinaremos tres aplicaciones reales: correo electrónico, World Wide Web y, finalmente, multimedia.

7.1 DNS—EL SISTEMA DE NOMBRES DE DOMINIO

Aunque en teoría los programas pueden hacer referencia a *hosts*, buzones de correo y otros recursos mediante sus direcciones de red (por ejemplo, IP), a las personas se les dificulta recordar estas direcciones. Además, enviar correo electrónico a *tana@128.111.24.41* significa que si el ISP u organización de Tana mueve el servidor de correo electrónico a una máquina diferente, la cual tiene una dirección IP diferente, la dirección de correo electrónico de Tana tiene que cambiar. Debido a esto, se introdujeron los nombres ASCII, con el fin de separar los nombres de máquina de las direcciones de máquina. De esta manera, la dirección de Tana podría ser algo como *tana@art.ucsb.edu*. Sin embargo, la red misma sólo comprende direcciones numéricas, por lo que se requieren algunos mecanismos para convertir las cadenas ASCII a direcciones de red. En las siguientes secciones analizaremos la forma en que se logra esta correspondencia en Internet.

Hace mucho, en los tiempos de ARPANET, sólo había un archivo, *hosts.txt*, en el que se listaban todos los *hosts* y sus direcciones IP. Cada noche, todos los *hosts* obtenían este archivo del sitio en el que se mantenía. En una red conformada por unas cuantas máquinas grandes de tiempo compartido, este método funcionaba razonablemente bien.

Sin embargo, cuando miles de estaciones de trabajo se conectaron a la red, todos se dieron cuenta de que este método no podría funcionar eternamente. Por una parte, el tamaño del archivo crecería de manera considerable. Un problema aún más importante era que ocurrirían conflictos constantes con los nombres de los *hosts* a menos de que dichos nombres se administraran centralmente, algo impensable en una red internacional enorme. Para resolver estos problemas, se inventó el **DNS** (**Sistema de Nombres de Dominio).**

La esencia del DNS es la invención de un esquema de nombres jerárquico basado en dominios y un sistema de base de datos distribuido para implementar este esquema de nombres. El DNS se usa principalmente para relacionar los nombres de *host* y destinos de correo electrónico con las direcciones IP, pero también puede usarse con otros fines. El DNS se define en los RFCs 1034 y 1035.

Muy brevemente, la forma en que se utiliza el DNS es la siguiente. Para relacionar un nombre con una dirección IP, un programa de aplicación llama a un procedimiento de biblioteca llamado **resolvedor,** y le pasa el nombre como parámetro. En la figura 6.6 vimos un ejemplo de un resolvedor, *gethostbyname*. El resolvedor envía un paquete UDP a un servidor DNS local, que después busca el nombre y devuelve la dirección IP al resolvedor, que entonces lo devuelve al solicitante. Una vez que tiene la dirección IP, el programa puede establecer una conexión TCP con el destino, o enviarle paquetes UDP.

7.1.1 El espacio de nombres del DNS

La administración de un grupo grande y continuamente cambiante de nombres es un problema nada sencillo. En el sistema postal, la administración de nombres se hace requiriendo letras para especificar (implícita o explícitamente) el país, estado o provincia, ciudad y calle, y dirección del destinatario. Con este tipo de direccionamiento jerárquico, no hay confusión entre el Marvin Anderson de Main St., en White Plains, N.Y. y el Marvin Anderson de Main St., en Austin, Texas. El DNS funciona de la misma manera.

Conceptualmente, Internet se divide en 200 **dominios** de nivel superior, cada uno de los cuales abarca muchos *hosts*. Cada dominio se divide en subdominios, los cuales, a su vez, también se dividen, y así sucesivamente. Todos estos dominios pueden representarse mediante un árbol, como se muestra en la figura 7-1. Las hojas del árbol representan los dominios que no tienen subdominios (pero que, por supuesto, contienen máquinas). Un dominio de hoja puede contener un solo *host*, o puede representar a una compañía y contener miles de *hosts*.

Los dominios de nivel superior se dividen en dos categorías: genéricos y de país. Los dominios genéricos originales son *com* (*comercial*), *edu* (instituciones educativas), *gov* (el gobierno federal de Estados Unidos), *int* (ciertas organizaciones internacionales), *mil* (las fuerzas armadas de Estados Unidos), *net* (proveedores de red) y *org* (organizaciones no lucrativas). Los dominios de país incluyen una entrada para cada país, como se define en la ISO 3166.

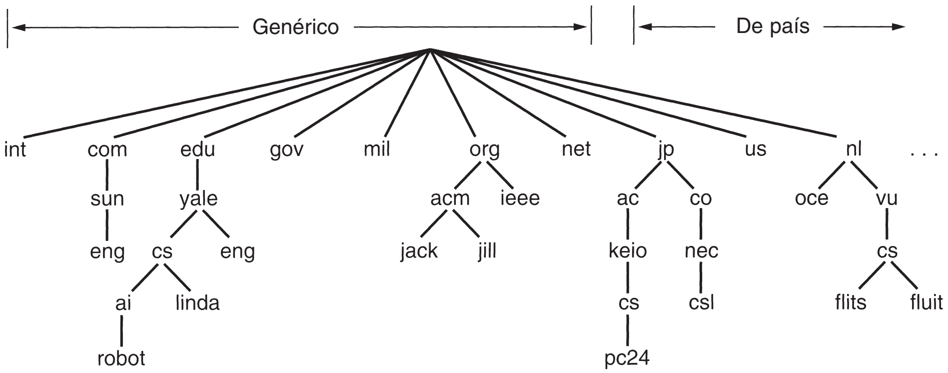


Figura 7-1. Parte del espacio de nombres de dominio de Internet.

En noviembre de 2000, ICANN aprobó cuatro nuevos dominios de nivel superior y propósito general: *biz* (negocios), *info* (información), *name* (nombres de personas) y *pro* (profesiones, como doctores y abogados). Además, se introdujeron otros tres nombres de dominio especializados de nivel superior a solicitud de ciertas industrias. Éstos son: *aero* (industria aeroespacial), *coop* (cooperativas) y *museum* (museos). En el futuro se agregarán otros dominios de nivel superior.

Como nota al margen, conforme Internet se vuelva más comercial, también se volverá más contenciosa. Por ejemplo, considere a *pro*. Estaba destinado para los profesionales certificados, pero, ¿quién es un profesional? y, ¿certificado por quién? Los doctores y abogados claramente son profesionales, pero, ¿qué pasa con los fotógrafos independientes, maestros de piano, magos, plomeros, barberos, exterminadores, artistas de tatuajes, soldados y prostitutas? ¿Estas ocupaciones son profesionales y, por lo tanto, candidatos a los dominios *pro*? De ser así, ¿quién certifica a los practicantes individuales?

En general, obtener un dominio de segundo nivel, como *nombre-de-compañía.*com, es fácil. Simplemente se necesita ir con el registrador del dominio de nivel superior correspondiente (*com*, en este caso) para ver si el nombre deseado está disponible y si no es la marca registrada de alguien más. Si no hay problemas, el solicitante paga una pequeña cuota anual y obtiene el nombre. En la actualidad, casi todas las palabras comunes (en inglés) ya se han tomado en el dominio *com*. Pruebe con artículos del hogar, animales, plantas, partes del cuerpo, etcétera. Hay muy pocos disponibles.

Cada dominio se nombra por la ruta hacia arriba desde él a la raíz (sin nombre). Los componentes se separan con puntos. Por lo tanto, el departamento de ingeniería de Sun Microsystems podría utilizar *eng.sun.com.*, en lugar de un nombre tipo UNIX como */com/sun/eng*. Observe que esta asignación jerárquica significa que *eng.sun.com.* no entra en conflicto con un uso potencial de *eng* —por ejemplo, *eng.yale.edu.,* que podría usarse en el departamento de inglés de Yale.

Los nombres de dominio pueden ser absolutos o relativos. Un nombre de dominio absoluto termina con un punto (por ejemplo, *eng.sun.com.*), y uno relativo no. Los nombres relativos tienen que interpretarse en algún contexto para determinar de manera única su significado verdadero. En

ambos casos, un dominio nombrado hace referencia a un nodo específico del árbol y a todos los nodos por debajo de él.

Los nombres de dominio no hacen distinción entre mayúsculas y minúsculas, por lo que *edu*, *Edu* y *EDU* significan lo mismo. Los nombres de componentes pueden ser de hasta 63 caracteres de longitud, y los de ruta completa de hasta 255 caracteres.

En principio, los dominios pueden introducirse en el árbol de dos maneras diferentes. Por ejemplo, *cs.yale.edu* también podría estar listado bajo el dominio de país *us* como *cs.yale.ct.us.* Sin embargo, en la práctica, casi todas las organizaciones de Estados Unidos están bajo un dominio genérico, y casi todas las de fuera de Estados Unidos están bajo el dominio de su país. No hay ninguna regla que impida registrarse bajo dos dominios de nivel superior, pero pocas organizaciones lo hacen, excepto las multinacionales (por ejemplo, *sony.com* y *sony.nl*).

Cada dominio controla cómo se asignan los dominios que están debajo de él. Por ejemplo, Japón tiene los dominios *ac.jp* y *co.jp* que son espejos de *edu* y *com.* Los Países Bajos no hacen esta distinción y ponen a todas las organizaciones directamente bajo *nl.* Por lo tanto, los siguientes tres son departamentos universitarios de informática:

1. *cs.yale.edu* (Universidad de Yale, en Estados Unidos)
2. *cs.vu.nl* (Vrije Universiteit, en los Países Bajos)
3. *cs.keio.ac.jp* (Universidad Keio, en Japón)

Para crear un nuevo dominio, se requiere el permiso del dominio en el que se incluirá. Por ejemplo, si se inicia un grupo VLSI en Yale y quiere que se le conozca como *vlsi.cs.yale.edu,* requiere permiso de quien administra *cs.yale.edu.* De la misma manera, si se crea una universidad nueva, digamos la Universidad del Norte de Dakota del Sur, debe solicitar al administrador del dominio *edu* que le asigne *unsd.edu.* De esta manera se evitan los conflictos de nombres y cada dominio puede llevar el registro de todos sus subdominios. Una vez que se ha creado y registrado un nuevo dominio puede crear subdominios, como *cs.unsd.edu,* sin obtener el permiso de nadie más arriba en el árbol.

Los nombres reflejan los límites organizacionales, no las redes físicas. Por ejemplo, si los departamentos de informática e ingeniería eléctrica se ubican en el mismo edificio y comparten la misma LAN, de todas maneras pueden tener dominios diferentes. De manera similar, si el departamento de informática está dividido entre el edificio Babbage y el edificio Turing, todos los *hosts* de ambos edificios pertenecerán, normalmente, al mismo dominio.

7.1.2 Registros de recursos

Cada dominio, sea un *host* individual o un dominio de nivel superior, puede tener un grupo de **registros de recursos** asociados a él. En un *host* individual, el registro de recursos más común es simplemente su dirección IP, pero también existen muchos otros tipos de registros de recursos. Cuando un resolvedor da un nombre de dominio al DNS, lo que recibe son los registros de

recursos asociados a ese nombre. Por lo tanto, la función real del DNS es relacionar los dominios de nombres con los registros de recursos.

Un registro de recursos tiene cinco tuplas. Aunque éstas se codifican en binario por cuestión de eficiencia, en la mayoría de las presentaciones, los registros de recursos se dan como texto ASCII, una línea por registro de recursos. El formato que usaremos es el siguiente:

Nombre_dominio Tiempo_de_vida Clase Tipo Valor

El *Nombre_dominio* indica el dominio al que pertenece este registro. Por lo general, existen muchos registros por dominio y cada copia de la base de datos contiene información de muchos dominios. Por lo tanto, este campo es la clave primaria de búsqueda usada para atender las consultas. El orden de los registros de la base de datos no es significativo.

El campo de *Tiempo_de_vida* es una indicación de la estabilidad del registro. La información altamente estable recibe un valor grande, como 86,400 (la cantidad de segundos en un día). La información altamente volátil recibe un valor pequeño, como 60 (1 minuto). Regresaremos a este punto después de haber estudiado el almacenamiento en caché.

El tercer campo de cada registro de recursos es *Class*. Para la información de Internet, siempre es *IN*. Para información que no es de Internet, se pueden utilizar otros códigos, pero en la práctica, éstos raramente se ven.

El campo *Tipo* indica el tipo de registro de que se trata. En la figura 7-2 se listan los tipos más importantes.

Tipo	Significado	Valor
SOA	Inicio de autoridad	Parámetros para esta zona
A	Dirección IP de un *host*	Entero de 32 bits
MX	Intercambio de correo	Prioridad, dominio dispuesto a aceptar correo electrónico
NS	Servidor de nombres	Nombre de un servidor para este dominio
CNAME	Nombre canónico	Nombre de dominio
PTR	Apuntador	Alias de una dirección IP
HINFO	Descripción del *host*	CPU y SO en ASCII
TXT	Texto	Texto ASCII no interpretado

Figura 7-2. Principales tipos de registro de recursos DNS.

Un registro *SOA* proporciona el nombre de la fuente primaria de información sobre la zona del servidor de nombres (que se describe más adelante), la dirección de correo electrónico de su administrador, un número de serie único y varias banderas y temporizadores.

El tipo de registro más importante es el registro *A* (dirección) que contiene una dirección IP de 32 bits de algún *host*. Cada *host* de Internet debe tener cuando menos una dirección IP, para que otras máquinas puedan comunicarse con él. Algunos *hosts* tienen dos o más conexiones de red, en cuyo caso tendrán un registro de recursos tipo *A* por cada conexión de red (y, por lo tanto, por

cada dirección IP). DNS se puede configurar para iterar a través de éstos, regresando el primer registro en la primera solicitud, el segundo registro en la segunda solicitud, y así sucesivamente.

El siguiente tipo de registro más importante es *MX*, que especifica el nombre del dominio que está preparado para aceptar correo electrónico del dominio especificado. Se utiliza porque no todas las máquinas están preparadas para aceptar correo electrónico. Si alguien desea enviar correo electrónico a, por ejemplo, *bill@microsoft.com*, el *host* emisor necesita encontrar un servidor de correo en *microsoft.com* que esté dispuesto a aceptar correo electrónico. El registro *MX* puede proporcionar esta información.

Los registros *NS* especifican servidores de nombres. Por ejemplo, cada base de datos DNS normalmente tiene un registro *NS* por cada dominio de nivel superior, de modo que el correo electrónico pueda enviarse a partes alejadas del árbol de nombres. Regresaremos a este punto más adelante.

Los registros *CNAME* permiten la creación de alias. Por ejemplo, una persona familiarizada con los nombres de Internet en general que quiere enviar un mensaje a alguien cuyo nombre de inicio de sesión es *paul* y que está en el departamento de informática del M.I.T., podría suponer que *paul@cs.mit.edu* funcionará. De hecho, esta dirección no funcionará, puesto que el dominio del departamento de informática del M.I.T. es *lcs.mit.edu.* Sin embargo, como servicio para la gente que no sabe esto, el M.I.T. podría crear una entrada *CNAME* para encaminar a la gente y a los programas en la dirección correcta. La entrada podría ser como la siguiente:

```
cs.mit.edu    86400    IN    CNAME    lcs.mit.edu
```

Al igual que *CNAME*, *PTR* apunta a otro nombre. Sin embargo, a diferencia de *CNAME*, que en realidad es sólo una definición de macro, *PTR* es un tipo de datos DNS normal, cuya interpretación depende del contexto en el que se encontró. En la práctica, *PTR* casi siempre se usa para asociar un nombre a una dirección IP a fin de permitir búsquedas de la dirección IP y devolver el nombre de la máquina correspondiente. Éstas se llaman **búsquedas invertidas.**

Los registros *HINFO* permiten que la gente conozca el tipo de máquina y sistema operativo al que corresponde un dominio. Por último, los registros *TXT* permiten a los dominios identificarse de modos arbitrarios. Ambos tipos de registro son para el provecho de los usuarios. Ninguno es obligatorio, por lo que los programas no pueden contar con que los recibirán (y probablemente no puedan manejarlos si los obtienen).

Por último, llegamos al campo *Valor*. Este campo puede ser un número, un nombre de dominio o una cadena ASCII. La semántica depende del tipo de registro. En la figura 7-2 se presenta una descripción corta de los campos de *Valor* para cada uno de los tipos principales de registro.

Como ejemplo del tipo de información que podría encontrarse en la base de datos DNS de un dominio, vea la figura 7-3. En ésta se ilustra una parte de una base de datos (semihipotética) correspondiente al dominio *cs.vu.nl* mostrado en la figura 7-1. La base de datos contiene siete tipos de registros de recursos.

La primera línea no de comentario de la figura 7-3 da un poco de información básica sobre el dominio, de lo que ya no nos ocuparemos. Las siguientes dos líneas dan información textual

```
;Datos autorizados correspondientes a cs.vu.nl
cs.vu.nl.       86400  IN  SOA    star boss (9527,7200,7200,241920,86400)
cs.vu.nl.       86400  IN  TXT    "Divisie Wiskunde en Informatica."
cs.vu.nl.       86400  IN  TXT    "Vrije Universiteit Amsterdam."
cs.vu.nl.       86400  IN  MX     1 zephyr.cs.vu.nl.
cs.vu.nl.       86400  IN  MX     2 top.cs.vu.nl.

flits.cs.vu.nl. 86400  IN  HINFO  Sun Unix
flits.cs.vu.nl. 86400  IN  A      130.37.16.112
flits.cs.vu.nl. 86400  IN  A      192.31.231.165
flits.cs.vu.nl. 86400  IN  MX     1 flits.cs.vu.nl.
flits.cs.vu.nl. 86400  IN  MX     2 zephyr.cs.vu.nl.
flits.cs.vu.nl. 86400  IN  MX     3 top.cs.vu.nl.
www.cs.vu.nl.   86400  IN  CNAME  star.cs.vu.nl
ftp.cs.vu.nl.   86400  IN  CNAME  zephyr.cs.vu.nl

rowboat                IN  A      130.37.56.201
                       IN  MX     1 rowboat
                       IN  MX     2 zephyr
                       IN  HINFO  Sun Unix

little-sister          IN  A      130.37.62.23
                       IN  HINFO  Mac MacOS

laserjet               IN  A      192.31.231.216
                       IN  HINFO  "HP Laserjet IIISi" Proprietary
```

Figura 7-3. Parte de una posible base de datos DNS para *cs.vu.nl.*

sobre la localización del dominio. Luego están dos entradas que dan el primer y segundo lugar a donde se intentará entregar correo electrónico dirigido a *person@cs.vu.nl.* Primero se intentará en la *zephyr* (una máquina específica). Si falla, a continuación se intentará en la *top*.

Después de la línea en blanco, que se agregó para hacer más clara la lectura, siguen líneas que indican que la *flits* es una estación de trabajo Sun que ejecuta UNIX, y dan sus dos direcciones IP. Después se indican tres posibilidades para manejar el correo electrónico enviado a *flits.cs.vu.nl.* La primera opción naturalmente es la *flits* misma, pero si está inactiva, la *zephyr* y la *top* son la segunda y tercera opciones, respectivamente. Luego viene un alias, *www.cs.vu.nl,* para que esta dirección pueda usarse sin designar una máquina específica. La creación de este alias permite a *cs.vu.nl* cambiar su servidor *Web* sin invalidar la dirección que la gente usa para llegar a él. Un argumento similar es válido para *ftp.cs.vu.nl.*

Las siguientes cuatro líneas contienen una entrada típica de una estación de trabajo, en este caso, *rowboat.cs.vu.nl.* La información proporcionada contiene la dirección IP, los destinos de correo primarios y secundarios, así como información sobre la máquina. Luego viene una entrada para un sistema no UNIX incapaz de recibir correo por sí mismo, seguida de una entrada para una impresora láser que está conectada a Internet.

Lo que no se muestra (y no está en este archivo) son las direcciones IP a usar para buscar los dominios de nivel superior. Éstas son necesarias para buscar *hosts* distantes pero, dado que no son parte del dominio *cs.vu.nl,* no están en este archivo. Tales direcciones son suministradas por los servidores raíz, cuyas direcciones IP están presentes en un archivo de configuración del sistema y se cargan en el caché del DNS cuando se arranca el servidor DNS. Hay aproximadamente una docena de servidores raíz esparcidos en todo el mundo, y cada uno conoce las direcciones IP de todos los servidores de dominio de nivel superior. Por lo tanto, si una máquina conoce la dirección IP de por lo menos un servidor raíz, puede buscar cualquier nombre DNS.

7.1.3 Servidores de nombres

Cuando menos en teoría, un solo servidor de nombres podría contener toda la base de datos DNS y responder a todas las consultas dirigidas a ella. En la práctica, este servidor estaría tan sobrecargado que sería inservible. Más aún, si llegara a caerse, la Internet completa se vendría abajo.

Para evitar los problemas asociados a tener una sola fuente de información, el espacio de nombres DNS se divide en **zonas** no traslapantes. En la figura 7-4 se muestra una manera posible de dividir el espacio de nombres de la figura 7-1. Cada zona contiene una parte del árbol y también contiene servidores de nombres que tienen la información de autorización correspondiente a esa zona. Por lo general, una zona tendrá un servidor de nombres primario, que obtiene su información de un archivo en su disco, y uno o más servidores de nombres secundarios, que obtienen su información del primario. Para mejorar la confiabilidad, algunos servidores de cierta zona pueden situarse fuera de la zona.

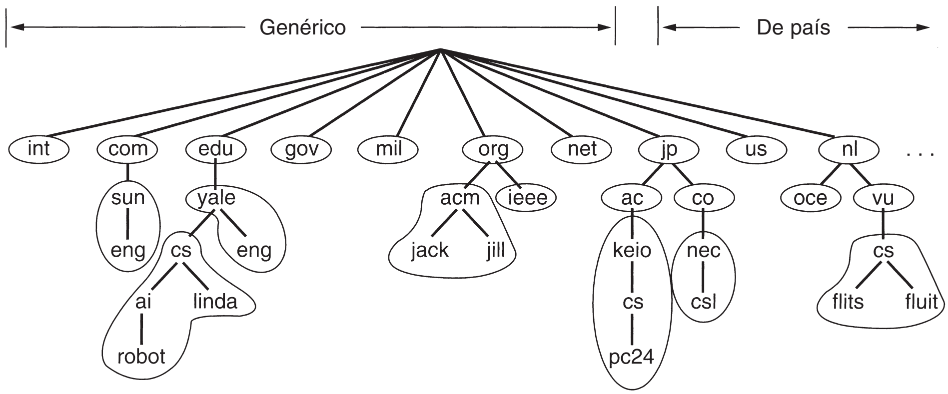


Figura 7-4. Parte del espacio de nombres DNS, donde se muestra la división en zonas.

El lugar donde se colocan los límites de las zonas dentro de una zona es responsabilidad del administrador de esa zona. Esta decisión se toma en gran medida con base en la cantidad de servidores de nombres deseados y su ubicación. Por ejemplo, en la figura 7-4, Yale tiene un servidor

para *yale.edu* que maneja *eng.yale.edu,* pero no *cs.yale.edu,* que es una zona aparte con sus propios servidores de nombres. Tal decisión podría tomarse cuando un departamento, como el de inglés, no desea operar su propio servidor de nombres, pero un departamento como el de informática sí. En consecuencia, *cs.yale.edu* es una zona aparte, pero *eng.yale.edu* no.

Cuando un resolvedor tiene una consulta referente a un nombre de dominio, la pasa a uno de los servidores de nombres locales. Si el dominio que se busca cae bajo la jurisdicción del servidor de nombres, como *ai.cs.yale.edu,* que cae bajo *cs.yale.edu,* devuelve los registros de recursos autorizados. Un **registro autorizado** es uno que proviene de la autoridad que administra el registro y, por lo tanto, siempre es correcto. Los registros autorizados contrastan con los registros en caché, que podrían no estar actualizados.

Por otro lado, si el dominio es remoto y no hay información disponible localmente sobre el dominio solicitado, el servidor de nombres envía un mensaje de consulta al servidor de nombres de nivel superior en el que le solicita dicho dominio. Para hacer más claro este proceso, considere el ejemplo de la figura 7-5. Aquí, un resolvedor de *flits.cs.vu.nl* quiere saber la dirección IP del *host linda.cs.yale.edu.* En el paso 1, envía una consulta al servidor de nombres local, *cs.vu.nl.* Esta consulta contiene el nombre de dominio buscado, el tipo (*A*) y la clase (*IN*).

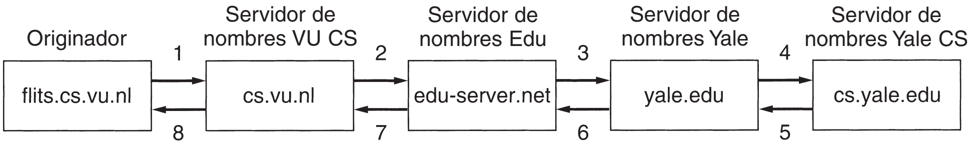


Figura 7-5. Manera en que un resolvedor busca un nombre remoto en ocho pasos.

Supongamos que al servidor de nombres local nunca se le ha solicitado este dominio, por lo que no sabe nada sobre él; puede preguntar a otros servidores de nombres cercanos, pero si ninguno de ellos sabe nada, enviará un paquete UDP al servidor de *edu* indicado en su base de datos (vea la figura 7-5), *edu-server.net.* Desde luego, es improbable que este servidor sepa la dirección de *linda.cs.yale.edu,* y probablemente tampoco sabe la de *cs.yale.edu,* pero debe conocer a todos sus hijos, por lo que reenvía la solicitud al servidor de nombres de *yale.edu* (paso 3). A su vez, éste reenvía la solicitud a *cs.yale.edu* (paso 4), que debe tener los registros de recursos autorizados. Puesto que cada solicitud es de un cliente a un servidor, el registro de recursos solicitado regresa a través de los pasos 5 a 8.

Una vez que estos registros regresan al servidor de nombres *cs.vu.nl,* se almacenan en caché, por si se necesitan posteriormente. Sin embargo, esta información no es autorizada, puesto que los cambios hechos en *cs.yale.edu* no se propagarán a todos los cachés del mundo que puedan saber sobre ella. Por esta razón, las entradas de caché no deben vivir demasiado tiempo. Ésta es la razón por la cual el campo *Tiempo_de_vida* se incluye en cada registro de recursos; indica a los servidores de nombres remotos cuánto tiempo deben mantener en caché los registros. Si cierta máquina ha tenido la misma dirección IP durante años, puede ser seguro poner en caché esa información

durante un día. En el caso de información más volátil, podría ser más seguro purgar los registros tras unos cuantos segundos o un minuto.

Vale la pena mencionar que el método de consultas aquí descrito se conoce como **consulta recursiva,** puesto que cada servidor que no tiene toda la información solicitada la busca en algún otro lado y luego la proporciona. Es posible un procedimiento alternativo. En él, cuando una consulta no puede satisfacerse localmente, ésta falla, pero se devuelve el nombre del siguiente servidor a intentar a lo largo de la línea. Algunos servidores no implementan consultas recursivas y siempre devuelven el nombre del siguiente servidor a intentar.

También vale la pena indicar que cuando un cliente DNS no recibe una respuesta antes de que termine su temporizador, por lo general probará con otro servidor la siguiente vez. La suposición aquí es que el servidor probablemente está inactivo, y no que la solicitud o la respuesta se perdieron.

Si bien DNS es muy importante para el funcionamiento correcto de Internet, todo lo que hace realmente es relacionar nombres simbólicos de máquinas con sus direcciones IP. No ayuda a localizar personas, recursos, servicios u objetos en general. Para localizar este tipo de cosas, se ha definido otro servicio de directorio, llamado **LDAP** (**Protocolo Ligero de Acceso al Directorio**)**.** Éste es una versión simplificada del servicio de directorio OSI X.500 y se describe en el RFC 2251. Organiza información como un árbol y permite búsquedas en componentes diferentes. Se puede considerar como un directorio telefónico de "páginas blancas". No lo analizaremos más en este libro, pero para mayor información, vea (Weltman y Dahbura, 2000).

7.2 CORREO ELECTRÓNICO

El correo electrónico, o ***e-mail*****,** como lo conocen muchos aficionados, ha existido por más de dos décadas. Antes de 1990, se utilizaba principalmente en ambientes académicos. En la década de 1990, se dio a conocer al público y creció en forma exponencial al punto que el número de mensajes de correo electrónico enviados por día ahora es mayor que el número de cartas por **correo caracol** (es decir, en papel).

El correo electrónico, como la mayoría de otras formas de comunicación, tiene sus propias convenciones y estilos. En particular, es muy informal y tiene un umbral bajo de uso. Las personas que nunca hubieran soñado con escribir una carta a un personaje importante, no dudarían un instante para enviarle un mensaje de correo electrónico.

El correo electrónico está lleno de abreviaturas, como BTW (*By The Way*, por cierto), ROTFL (*Rolling On The Floor Laughing*, rodando por el suelo muerto de risa) e IMHO (*In My Humble Opinión*, en mi humilde opinión). Muchas personas también utilizan pequeños símbolos ASCII llamados **caritas** o **símbolos de emociones** en sus mensajes de correo electrónico. Algunos de los más interesantes se listan en la figura 7-6. Para la mayoría, rote el libro 90 grados en dirección de las manecillas del reloj y se verán con mayor claridad. Un minilibro que presenta cerca de 650 caritas es (Sanderson y Dougherty, 1993).

Los primeros sistemas de correo electrónico simplemente consistían en protocolos de transferencia de archivos, con la convención de que la primera línea de cada mensaje (es decir, archivo)

Emoticono	Significado	Emoticono	Significado	Emoticono	Significado
:-)	Estoy feliz	=|:-)	Abe Lincoln	:+)	Nariz grande
:-(	Estoy triste/enojado	=):-)	Tío Sam	:-))	Realmente feliz
:-|	Estoy apático	*<:-)	Santa Claus	:-{)	Con bigote
;-)	Estoy guiñando un ojo	<:-(	Bobo	#:-)	Peinado yuppy
:-(O)	Estoy gritando	(-:	Zurdo	8-)	Con gafas
:-(*)	Estoy vomitando	:-)X	Hombre con moño	C:-)	Inteligente

Figura 7-6. Algunos emoticonos. No estarán en el examen final :-)

contenía la dirección del destinatario. A medida que pasó el tiempo, las limitaciones de este enfoque se hicieron obvias. Algunas de las quejas eran:

1. El envío de un mensaje a un grupo de personas era laborioso. Los administradores con frecuencia necesitaban esta facilidad para enviar memorandos a todos sus subordinados.
2. Los mensajes no tenían estructura interna, lo que dificultaba el proceso por computadora. Por ejemplo, si un mensaje reenviado se incluía en el cuerpo de otro mensaje, era difícil extraer la parte reenviada del mensaje recibido.
3. El originador (remitente) nunca sabía si el mensaje había llegado o no.
4. Si alguien planeaba ausentarse por un viaje de negocios durante varias semanas y quería que todo el correo entrante fuera manejado por su secretaria, esto no era fácil de arreglar.
5. La interfaz de usuario estaba mal integrada al sistema de transmisión, pues requería que el usuario primero editara un archivo, luego saliera del editor e invocara el programa de transferencia de archivos.
6. No era posible crear y enviar mensajes que contuvieran una mezcla de texto, dibujos, facsímiles y voz.

A medida que se acumuló experiencia, se propusieron sistemas de correo electrónico más elaborados. En 1982, se publicaron las propuestas de correo electrónico de ARPANET como el RFC 821 (protocolo de transmisión) y el RFC 822 (formato de mensaje). Las revisiones menores, los RFCs 2821 y 2822, se han vuelto estándares de Internet, pero todas las personas aún se refieren al correo electrónico de Internet como el RFC 822.

En 1984, el CCITT redactó la recomendación X.400. Después de dos décadas de competencia, los sistemas de correo electrónico basados en el RFC 822 se utilizan ampliamente, mientras que aquellos basados en X.400 han desaparecido. El hecho de que un sistema creado por un grupo de estudiantes de licenciatura en ciencias de la computación haya vencido a un estándar internacional que estaba fuertemente respaldado por todos los PTTs en el mundo, muchos gobiernos y una parte significativa de la industria de la computación nos recuerda a la historia bíblica de David y Goliath.

La razón del éxito del RFC 822 no es que sea tan bueno, sino que X.400 estaba tan pobremente diseñado y era tan complejo que nadie podía implementarlo bien. Dada la opción entre un sistema de correo electrónico basado en el RFC 822 simple, pero funcional, y un sistema de correo electrónico X.400 supuestamente maravilloso, aunque complejo, la mayoría de las organizaciones eligen el primero. Tal vez aquí haya una lección. En consecuencia, nuestro análisis del correo electrónico se concentrará en el sistema de correo electrónico de Internet.

7.2.1 Arquitectura y servicios

En esta sección ofreceremos un panorama de lo que pueden hacer los sistemas de correo electrónico y la manera en que están organizados. Normalmente consisten en dos subsistemas: los **agentes de usuario,** que permiten a la gente leer y enviar correo electrónico, y los **agentes de transferencia de mensajes,** que mueven los mensajes del origen al destino. Los agentes de usuario son programas locales que proporcionan un método basado en comandos, en menús o en una interfaz gráfica para interactuar con el sistema de correo electrónico. Los agentes de transferencia de mensajes son por lo común **demonios** (***daemons***) del sistema que operan en segundo plano y mueven correo electrónico a través del sistema.

Por lo general, los sistemas de correo electrónico desempeñan cinco funciones básicas, como se describe a continuación.

La **redacción** se refiere al proceso de crear mensajes y respuestas. Aunque es posible utilizar cualquier editor de texto para el cuerpo del mensaje, el sistema mismo puede proporcionar asistencia con el direccionamiento y los numerosos campos de encabezado que forman parte de cada mensaje. Por ejemplo, al contestar un mensaje, el sistema de correo electrónico puede extraer la dirección del remitente e insertarla en forma automática en el lugar adecuado de la respuesta.

La **transferencia** se refiere a mover mensajes del remitente al destinatario. En gran medida, esto requiere establecer una conexión con el destino o alguna máquina intermedia, enviar el mensaje y liberar la conexión. El sistema de correo electrónico debe hacer esto en forma automática, sin molestar al usuario.

La **generación del informe** tiene que ver con indicar al remitente lo que ocurrió con el mensaje: ¿se entregó, se rechazó o se perdió? Existen muchas aplicaciones en las que la confirmación de la entrega es importante y puede incluso tener una aplicación legal ("Bien, Su Señoría, mi sistema de correo electrónico no es muy confiable, por lo que creo que la citación electrónica se perdió en algún lado").

La **visualización** de los mensajes de entrada es necesaria para que la gente pueda leer su correo electrónico. A veces se requiere cierta conversión o debe invocarse un visor especial, por ejemplo, si el mensaje es un archivo PostScript o voz digitalizada. Algunas veces también se intentan conversiones y formateo sencillos.

La **disposición** es el paso final y tiene que ver con lo que el destinatario hace con el mensaje una vez que lo recibe. Las posibilidades incluyen tirarlo antes de leerlo, desecharlo después de leerlo, guardarlo, etcétera. También debe ser posible recuperar y releer mensajes guardados, reenviarlos o procesarlos de otras maneras.

Además de estos servicios básicos, la mayoría de los sistemas de correo electrónico, especialmente los corporativos internos, proporcionan una gran variedad de características avanzadas. Brevemente mencionaremos algunas de éstas. Cuando la gente se muda, o cuando está lejos durante cierto tiempo, podría querer el reenvío de su correo electrónico, por lo que el sistema debe ser capaz de hacer esto de manera automática.

La mayoría de los sistemas permite a los usuarios crear **buzones de correo** para almacenar el correo entrante. Se requieren comandos para crear y destruir buzones de correo, inspeccionar el contenido de los buzones, insertar y borrar mensajes de los buzones, etcétera.

Los gerentes corporativos con frecuencia necesitan enviar un mensaje a cada uno de sus subordinados, clientes o proveedores. Esto da pie al concepto de **lista de correo,** que es una lista de direcciones de correo electrónico. Cuando se envía un mensaje a la lista de correo, se entregan copias idénticas a todos los de la lista.

Otras características avanzadas son copias ocultas, correo electrónico de alta prioridad, correo electrónico secreto (es decir, encriptado), destinatarios alternos si el primario no está disponible y la capacidad de que las secretarias manejen el correo electrónico de su jefe.

En la actualidad, el correo electrónico se usa ampliamente en la industria para la comunicación intracompañía. Permite que los empleados remotos cooperen en proyectos complejos, incluso si están en otros husos horarios. Al eliminar la mayoría de las señales asociadas al rango, edad y género, los debates realizados por correo electrónico tienden a enfocarse principalmente en las ideas, no en el *status* corporativo. Con el correo electrónico, una idea brillante de un estudiante becario puede tener más impacto que una idea tonta de un vicepresidente ejecutivo.

Una idea clave de todos los sistemas modernos de correo electrónico es la distinción entre el **sobre** y su contenido. El sobre encapsula el mensaje; contiene toda la información necesaria para transportar el mensaje, como dirección de destino, prioridad y nivel de seguridad, la cual es diferente del mensaje mismo. Los agentes de transporte del mensaje usan el sobre para enrutar, al igual que la oficina postal.

El mensaje dentro del sobre contiene dos partes: el **encabezado** y el **cuerpo.** El encabezado contiene información de control para los agentes de usuario. El cuerpo es por completo para el destinatario humano. Los sobres y mensajes se ilustran en la figura 7-7.

7.2.2 El agente de usuario

Los sistemas de correo electrónico tienen dos partes básicas, como hemos visto: los agentes de usuario y los agentes de transferencia de mensajes. En esta sección veremos a los agentes de usuario. Un agente de usuario normalmente es un programa (a veces llamado lector de correo) que acepta una variedad de comandos para redactar, recibir y contestar los mensajes, así como para manipular los buzones de correo. Algunos agentes de usuario tienen una interfaz elegante operada por menús o por iconos que requiere un ratón, mientras que otros esperan comandos de un carácter desde el teclado. Funcionalmente, ambos son iguales. Algunos sistemas son manejados por menús o por iconos, pero también tienen métodos abreviados de teclado.

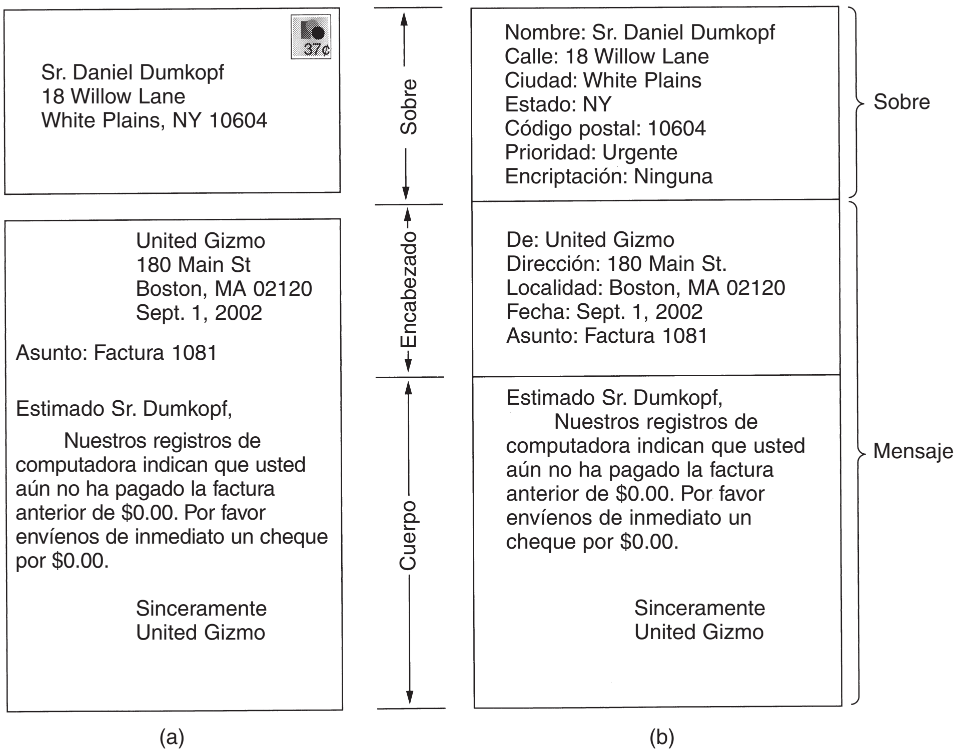


Figura 7-7. Sobres y mensajes. (a) Correo postal. (b) Correo electrónico.

Envío de correo electrónico

Para enviar un mensaje de correo electrónico, el usuario debe proporcionar el mensaje, la dirección de destino y, posiblemente, algunos otros parámetros. El mensaje puede producirse con un editor de texto independiente, un programa de procesamiento de texto o, posiblemente, un editor de texto incorporado en el agente de usuario. La dirección de destino debe estar en un formato que el agente de usuario pueda manejar. Muchos agentes de usuario esperan direcciones DNS de la forma *usuario@dirección-dns*. Dado que las hemos estudiado antes, no repetiremos ese material aquí.

Sin embargo, vale la pena indicar que existen otras formas de direccionamiento. En particular, las direcciones X.400 tienen un aspecto radicalmente diferente del de las direcciones DNS; se componen de pares *atributo* = *valor*, separadas por diagonales, por ejemplo,

/C=US/ST=MASSACHUSETTS/L=CAMBRIDGE/PA=360 MEMORIAL DR./CN=KEN SMITH/

Esta dirección especifica un país, estado, localidad, dirección personal y nombre común (Ken Smith). Son posibles muchos otros atributos, de modo que puede enviarse correo electrónico a alguien cuyo nombre no se conoce, siempre y cuando se conozca un número suficiente de los otros

atributos (por ejemplo, compañía y puesto). Aunque los nombres X.400 son considerablemente menos convenientes que los DNS, la mayoría de los sistemas de correo electrónico tienen alias (algunas veces llamados sobrenombres) que permiten que los usuarios introduzcan o seleccionen un nombre de una persona y obtengan la dirección de correo electrónico correcta. En consecuencia, incluso con direcciones X.400, por lo general no es necesario teclear realmente estas extrañas cadenas.

La mayoría de los sistemas de correo electrónico manejan listas de correo, por lo que un usuario puede enviar el mismo mensaje a un grupo de personas con un solo comando. Si la lista de correo se mantiene localmente, el agente usuario puede sencillamente enviar un mensaje por separado a cada destinatario. Sin embargo, si la lista se mantiene de manera remota, los mensajes se multiplicarán ahí. Por ejemplo, si un grupo de observadores de pájaros tiene una lista de correo llamada *birders* instalada en *meadowlark.arizona.edu,* entonces cualquier mensaje enviado a *birders-@meadowlark.arizona.edu* se enrutará a la Universidad de Arizona y se multiplicará ahí para producir mensajes individuales para todos los miembros de la lista de correo, sin importar el lugar del mundo en el que estén. Los usuarios de esta lista de correo no pueden saber que es una lista de correo. Podría ser simplemente el buzón de correo personal del profesor Gabriel O. Birders.

Lectura del correo electrónico

Por lo común, cuando se inicia un agente de usuario, buscará en el buzón del usuario el correo electrónico recibido, antes de presentar otra cosa en la pantalla. Después puede anunciar la cantidad de mensajes en el buzón o presentar un resumen de una línea de cada uno y esperar un comando.

Como ejemplo del funcionamiento de un agente de usuario, veamos una situación típica de correo. Después de iniciar el agente de usuario, el usuario solicita un resumen de su correo electrónico. A continuación aparece una pantalla como la de la figura 7-8. Cada línea hace referencia a un mensaje. En este ejemplo, el buzón contiene ocho mensajes.

#	Banderas	Bytes	Remitente	Asunto
1	K	1030	asw	Cambios a MINIX
2	KA	6348	trudy	No todas las Trudies son molestas
3	K F	4519	Amy N. Wong	Solicitud de información
4		1236	bal	Bioinformática
5		104110	kaashoek	Material sobre las redes de igual a igual
6		1223	Frank	Re: Revisará una propuesta de otorgamiento
7		3110	guido	Nuestro documento ha sido aceptado
8		1204	dmr	Re: Visita de mis estudiantes

Figura 7-8. Ejemplo de exhibición del contenido de un buzón.

Cada línea de la pantalla contiene varios campos extraídos del sobre o del encabezado del mensaje correspondiente. En un sistema sencillo de correo electrónico, la selección de campos a presentar está incorporada en el programa. En uno más refinado, el usuario puede especificar los campos a presentar proporcionando un **perfil de usuario,** un archivo que describe el formato de

presentación. En este ejemplo, el primer campo es el número de mensaje. El segundo campo, *Banderas*, puede contener una *K*, lo que significa que el mensaje no es nuevo, sino que ha sido leído previamente y guardado en el buzón; una *A*, lo que quiere decir que el mensaje ya ha sido contestado; y/o una *F*, lo que indica que ese mensaje ha sido reenviado a alguien más. También son posibles otras banderas.

El tercer campo indica la longitud del mensaje y el cuarto indica quién envió el mensaje. Puesto que este campo simplemente se extrae del mensaje, puede contener nombres de pila, nombres completos, iniciales, claves de acceso o cualquier otra cosa que el remitente decida poner ahí. Por último, el campo de *Asunto* da un resumen breve del tema del mensaje. La gente que no incluye un campo de *Asunto* con frecuencia descubre que las respuestas a su correo electrónico tienden a no recibir la prioridad más alta.

Una vez que se han visualizado los encabezados, el usuario puede realizar cualquiera de diversas acciones, como desplegar o eliminar el mensaje. Los sistemas más antiguos se basaban en texto y típicamente utilizaban comandos de un carácter para realizar estas tareas, como T (teclear mensaje), A (responder a mensaje), D (borrar mensaje) y F (reenviar mensaje). Un argumento especificaba el mensaje en cuestión. Los sistemas más recientes utilizan interfaces gráficas. Por lo general, el usuario selecciona un mensaje con el ratón y después hace clic en un icono para escribir, responder a él, eliminarlo o reenviarlo.

El correo electrónico ha recorrido un largo camino desde los días cuando era simplemente transferencia de archivos. Los agentes de usuario refinados hacen posible manejar un gran volumen de correo electrónico. Para las personas que reciben y envían miles de mensajes de correo electrónico al año, tales herramientas son invaluables.

7.2.3 Formatos de mensaje

Pasemos ahora de la interfaz de usuario al formato de los mensajes de correo electrónico mismos. Primero veremos el correo electrónico ASCII básico usando el RFC 822. Después de eso, veremos las extensiones multimedia del RFC 822.

RFC 822

Los mensajes consisten en un sobre primitivo (descrito en el RFC 821), algunos campos de encabezado, una línea en blanco y el cuerpo del mensaje. Cada campo de encabezado consiste (lógicamente) en una sola línea de texto ASCII que contiene el nombre del campo, dos puntos (:) y, para la mayoría de los campos, un valor. El RFC 822 es un estándar viejo y no distingue claramente los campos del sobre de los de encabezado. Aunque se revisó en el RFC 2822, no fue posible restaurarlo por completo debido a su uso amplio. En el uso normal, el agente de usuario construye un mensaje y lo pasa al agente de transferencia de mensajes, quien después usa algunos campos del encabezado para construir el sobre, una mezcla un tanto anticuada de mensaje y sobre.

En la figura 7-9 se listan los principales campos de encabezado relacionados con el transporte del mensaje. El campo *To:* indica la dirección DNS del destinatario primario. Está permitido tener

varios destinatarios. El campo *Cc:* indica la dirección de los destinatarios secundarios. En términos de entrega, no hay diferencia entre los destinatarios primarios y los secundarios. Es una diferencia por entero psicológica que puede ser importante para los participantes, pero que no lo es para el sistema de correo. El término *Cc:* (copia al carbón) es un poco anticuado, puesto que las computadoras no usan papel carbón, pero está muy establecido. El campo *Bcc:* (copia oculta) es como el campo *Cc:*, excepto que esta línea se borra de todas las copias enviadas a los destinatarios primarios y secundarios. Esta característica permite a la gente mandar copias a terceros sin que los destinatarios primarios y secundarios lo sepan.

Encabezado	Significado
To:	Direcciones de correo electrónico de los destinatarios primarios
Cc:	Direcciones de correo electrónico de los destinatarios secundarios
Bcc:	Direcciones de correo electrónico para las copias ocultas
From:	Persona o personas que crearon el mensaje
Sender:	Dirección de correo electrónico del remitente
Received:	Línea agregada por cada agente de transferencia en la ruta
Return-Path:	Puede usarse para identificar una ruta de regreso al remitente

Figura 7-9. Campos de encabezado RFC 822 relacionados con el transporte de mensajes.

From: y *Sender:* indican quién escribió y envió el mensaje, respectivamente; pueden ser diferentes. Por ejemplo, un ejecutivo de negocios puede escribir un mensaje, pero su secretaria podría ser la que lo transmita. En este caso, el ejecutivo estaría listado en el campo *From:* y la secretaria en el campo *Sender:*. El campo *From:* es necesario, pero el campo *Sender:* puede omitirse si es igual al campo *From:*. Estos campos son necesarios en caso de que el mensaje no pueda entregarse y deba devolverse al remitente.

Se agrega una línea que contiene *Received:* por cada agente de transferencia de mensajes en el camino. La línea contiene la identidad del agente, la fecha y hora de recepción del mensaje, y otra información que puede servir para encontrar fallas en el sistema de enrutamiento.

El campo *Return-Path:* es agregado por el agente de transferencia de mensajes final y estaba destinado para indicar la manera de regresar al remitente. En teoría, esta información puede tomarse de todos los encabezados *Received:* (con excepción del nombre del buzón del remitente), pero pocas veces se llena de esta manera y por lo general contiene sólo la dirección del remitente.

Además de los campos de la figura 7-9, los mensajes RFC 822 también pueden contener una variedad de campos de encabezado usados por los agentes de usuario o los destinatarios. En la figura 7-10 se listan los más comunes. La mayoría de ellos autoexplicativos, por lo que no los veremos en detalle.

El campo *Reply-To:* a veces se usa cuando ni la persona que redactó el mensaje ni la que lo envió quieren ver la respuesta. Por ejemplo, un gerente de *marketing* escribe un mensaje de correo electrónico para informar a los clientes sobre un producto nuevo. El mensaje lo envía una

Encabezado	Significado
Date:	Fecha y hora de envío del mensaje
Reply-To:	Dirección de correo electrónico a la que deben enviarse las contestaciones
Message-Id:	Número único para referencia posterior a este mensaje
In-Reply-To:	Identificador del mensaje al que éste responde
References:	Otros identificadores de mensaje pertinentes
Keywords:	Claves seleccionadas por el usuario
Subject:	Resumen corto del mensaje para desplegar en una línea

Figura 7-10. Algunos campos usados en el encabezado de mensaje RFC 822.

secretaria, pero el campo *Reply-To:* indica el jefe del departamento de ventas, quien puede contestar preguntas y surtir pedidos. Este campo también es útil cuando el emisor tiene dos cuentas de correo electrónico y desea que la respuesta llegue a su otra cuenta.

El documento RFC 822 indica explícitamente que los usuarios pueden inventar encabezados nuevos para uso privado, siempre y cuando comiencen con la cadena *X*-. Se garantiza que no habrá encabezados futuros que usen nombres que comiencen con *X*-, a fin de evitar conflictos entre los encabezados oficiales y los privados. A veces algunos estudiantes universitarios, pasándose de listos incluyen campos como *X-Fruta-del-Día:* o *X-Enfermedad-de-la-Semana:,* que son legales, aunque no muy ilustrativos.

Tras los encabezados viene el cuerpo del mensaje. Los usuarios pueden poner aquí lo que les venga en gana. Algunos terminan sus mensajes con firmas complicadas, incluidas caricaturas sencillas en ASCII, citas de autoridades mayores y menores, pronunciamientos políticos y renuncias de responsabilidades de todo tipo (por ejemplo, la corporación ABC no es responsable de mis opiniones; ni siquiera las puede entender).

MIME—Extensiones Multipropósito de Correo Internet

En los primeros días de ARPANET, el correo electrónico consistía exclusivamente en mensajes de texto escritos en inglés y expresados en ASCII. En tal entorno, el RFC 822 hacía todo el trabajo: especificaba los encabezados, pero dejaba el contenido en manos del usuario. Hoy día, en la red mundial de Internet, este método ya no es adecuado. Los problemas incluyen envío y recepción de:

1. Mensajes en idiomas con acentos (por ejemplo, español, francés y alemán).
2. Mensajes en alfabetos no latinos (por ejemplo, hebreo y ruso).
3. Mensajes en idiomas sin alfabetos (por ejemplo, chino y japonés).
4. Mensajes que no contienen texto (por ejemplo, audio y vídeo).

Se propuso una solución en el RFC 1341 y se actualizó en los RFCs 2045-2049. Esta solución, llamada **MIME** (**Extensiones Multipropósito de Correo Internet**), se usa ampliamente. A continuación la describiremos. Para información adicional sobre MIME, vea dichos RFCs.

La idea básica de MIME es continuar usando el formato RFC 822, pero agregar una estructura al cuerpo del mensaje y definir reglas de codificación para los mensajes no ASCII. Al no desviarse del 822, los mensajes MIME pueden enviarse usando los programas y protocolos de correo electrónico existentes. Todo lo que tiene que cambiarse son los programas emisores y receptores, lo que pueden hacer los usuarios mismos.

MIME define cinco nuevos encabezados de mensaje, como se muestra en la figura 7-11. El primero de éstos simplemente indica al agente de usuario receptor del mensaje que está tratando con un mensaje MIME, así como la versión del MIME que usa. Se considera que cualquier mensaje que no contenga un encabezado *MIME-Version:* es un mensaje de texto normal en inglés, y se procesa como tal.

Encabezado	Significado
MIME-Version:	Identifica la versión de MIME
Content-Description:	Cadena de texto que describe el contenido
Content-Id:	Identificador único
Content-Transfer-Encoding:	Cómo se envuelve el mensaje para su transmisión
Content-Type:	Naturaleza del mensaje

Figura 7-11. Encabezados RFC 822 agregados por MIME.

El encabezado *Content-Description:* es una cadena ASCII que dice lo que está en el mensaje. Este encabezado es necesario para que el destinatario sepa si vale la pena descodificar y leer el mensaje. Si la cadena dice: “Foto del jerbo de Bárbara” y la persona que recibe el mensaje no es un gran fanático de los jerbos, el mensaje probablemente será descartado en lugar de descodificado para dar una foto a color de alta definición.

El encabezado *Content-Id:* identifica el contenido; usa el mismo formato que el encabezado estándar *Message-Id:*.

Content-Transfer-Encoding: indica la manera en que está envuelto el cuerpo para su transmisión a través de una red donde se podría tener problemas con la mayoría de los caracteres distintos de las letras, números y signos de puntuación. Se proporcionan cinco esquemas (más un escape hacia nuevos esquemas). El esquema más sencillo es simplemente texto ASCII. Los caracteres ASCII usan 7 bits y pueden transportarse directamente mediante el protocolo de correo electrónico siempre y cuando ninguna línea exceda 1000 caracteres.

El siguiente esquema más sencillo es lo mismo, pero utiliza caracteres de 8 bits, es decir, todos los valores de 0 a 255. Este esquema de codificación viola el protocolo (original) del correo electrónico de Internet, pero es usado por algunas partes de Internet que implementan ciertas extensiones al protocolo original. Mientras que la declaración de la codificación no la hace legal,

hacerla explícita puede cuando menos explicar las cosas cuando algo sucede mal. Los mensajes que usan codificación de 8 bits deben aún adherirse a la longitud máxima de línea estándar.

Peor aún son los mensajes que usan codificación binaria. Éstos son archivos binarios arbitrarios que no sólo utilizan los 8 bits, sino que ni siquiera respetan el límite de 1000 caracteres por línea. Los programas ejecutables caen en esta categoría. No se da ninguna garantía de que los mensajes en binario llegarán correctamente, pero mucha gente los envía de todos modos.

La manera correcta de codificar mensajes binarios es usar **codificación base64,** a veces llamada **armadura ASCII.** En este esquema se dividen grupos de 24 bits en unidades de 6 bits, y cada unidad se envía como un carácter ASCII legal. La codificación es "A" para 0, "B" para 1, etcétera, seguidas por las 26 letras minúsculas, los 10 dígitos y, por último, + y / para el 62 y 63, respectivamente. Las secuencias == e = se usan para indicar que el último grupo contenía sólo 8 o 16 bits, respectivamente. Los retornos de carro y avances de línea se ignoran, por lo que puede introducirse a voluntad para mantener la línea lo suficientemente corta. Puede enviarse un texto binario arbitrario usando este esquema.

En el caso de mensajes que son casi completamente ASCII, pero con algunos caracteres no ASCII, la codificación base 64 es algo ineficiente. En su lugar se usa una codificación conocida como **codificación entrecomillada imprimible.** Simplemente es ASCII de 7 bits, con todos los caracteres por encima de 127 codificados como un signo de igual seguido del valor del carácter en dos dígitos hexadecimales.

En resumen, los datos binarios deben enviarse codificados en base 64 o en forma entrecomillada imprimible. Cuando hay razones válidas para no usar uno de estos esquemas, es posible especificar una codificación definida por el usuario en el encabezado *Content-Transfer-Encoding:*.

El último encabezado mostrado en la figura 7-11 en realidad es el más interesante. Especifica la naturaleza del cuerpo del mensaje. En el RFC 2045 hay siete tipos definidos, cada uno de los cuales tiene uno o más subtipos. El tipo y el subtipo se separan mediante una diagonal, como en

```
Content-Type: video/mpeg
```

El subtipo debe indicarse de manera explícita en el encabezado; no se proporcionan valores predeterminados. La lista inicial de tipos y subtipos especificada en el RFC 2045 se da en la figura 7-12. Se han agregado muchos otros desde entonces, y se agregan entradas nuevas todo el tiempo, a medida que surge la necesidad.

Veamos brevemente la lista de tipos. El tipo *text* es para texto normal. La combinación *text/plain* es para mensajes ordinarios que pueden visualizarse como se reciben, sin codificación ni ningún procesamiento posterior. Esta opción permite el transporte de mensajes ordinarios en MIME con sólo unos pocos encabezados extra.

El subtipo *text/enriched* permite la inclusión de un lenguaje de marcado sencillo en el texto. Este lenguaje proporciona una manera independiente del sistema para indicar negritas, cursivas, tamaños en puntos menores y mayores, sangrías, justificaciones, sub y superíndices, y formatos sencillos de página. El lenguaje de marcado se basa en SGML, el lenguaje estándar genérico de

Tipo	Subtipo	Descripción
Texto	Plano	Texto sin formato
	Enriquecido	Texto con comandos de formato sencillos
Imagen	Gif	Imagen fija en formato GIF
	Jpeg	Imagen fija en formato JPEG
Audio	Básico	Sonido
Vídeo	Mpeg	Película en formato MPEG
Aplicación	Octet-stream	Secuencia de bytes no interpretada
	Postscript	Documento imprimible en PostScript
Mensaje	Rfc822	Mensaje MIME RFC 822
	Parcial	Mensaje dividido para su transmisión
	Externo	El mensaje mismo debe obtenerse de la red
Multipartes	Mezclado	Partes independientes en el orden especificado
	Alternativa	Mismo mensaje en diferentes formatos
	Paralelo	Las partes deben verse en forma simultánea
	Compendio	Cada parte es un mensaje RFC 822 completo

Figura 7-12. Tipos y subtipos MIME definidos en el RFC 2045.

etiquetas que también se utiliza como la base del HTML de World Wide Web. Por ejemplo, el mensaje

Ha llegado el <bold> momento </bold>, dijo la <italic> morsa </italic>...

se presentaría como

Ha llegado el **momento,** dijo la *morsa*...

Es responsabilidad del sistema receptor seleccionar la presentación adecuada. Si hay negritas y cursivas disponibles, pueden usarse; de otra manera, pueden usarse colores, parpadeos, vídeo inverso, etcétera, como énfasis. Los diferentes sistemas pueden hacer selecciones distintas, lo cual hacen en realidad.

Cuando Web se volvió popular, se agregó un nuevo subtipo *texto/html* (en el RFC 2854) para permitir que las páginas Web se enviaran en el correo electrónico del RFC 822. En el RFC 3023 se define un subtipo para el lenguaje de marcado extensible, *texto/xml*. Analizaremos HTML y XML más adelante en este capítulo.

El siguiente tipo MIME es *imagen*, que se usa para transmitir imágenes fijas. Hoy día se usan muchos formatos para almacenar y transmitir imágenes, tanto con compresión como sin ella. Dos de éstos, GIF y JPEG, están integrados en la mayoría de los navegadores, pero también existen muchos otros, los cuales se han agregado a la lista original.

Los tipos *audio* y *vídeo* son para sonido e imágenes en movimiento, respectivamente. Observe que *vídeo* incluye sólo la información visual, no la pista de sonido. Si se debe transmitir una película con sonido, tal vez sea necesario transmitir las partes de vídeo y de audio por separado, dependiendo del sistema de codificación usado. El primer formato de vídeo definido fue el diseñado por el grupo de modesto nombre MPEG (Grupo de Expertos en Imágenes en Movimiento), pero desde entonces se han agregado otros. Además de *audio/basic*, se agregó un nuevo tipo de audio, *audio/mpeg*, en el RFC 3003 para permitir que las personas pudieran enviar archivos de audio MP3.

El tipo *aplicación* es un tipo general para los formatos que requieren procesamiento externo no cubierto por ninguno de los otros tipos. Un *octet-stream* simplemente es una secuencia de bytes no interpretados. Al recibir una de tales cadenas, un agente de usuario probablemente debería presentarla en pantalla sugiriendo al usuario que lo copie en un archivo y solicitándole un nombre de archivo. El procesamiento posterior es responsabilidad del usuario.

Otro subtipo definido es *postscript,* que se refiere al lenguaje PostScript producido por Adobe Systems y ampliamente usado para describir páginas impresas. Muchas impresoras tienen intérpretes PostScript integrados. Aunque un agente de usuario simplemente puede llamar a un intérprete PostScript externo para visualizar los archivos PostScript entrantes, hacerlo no está exento de riesgos. PostScript es un lenguaje de programación completo. Con el tiempo necesario, una persona lo bastante masoquista podría escribir un compilador de C o un sistema de administración de base de datos en PostScript. El despliegue de un mensaje PostScript entrante puede hacerse ejecutando el programa PostScript contenido en él. Además de desplegar algo de texto, este programa puede leer, modificar o borrar los archivos del usuario, así como tener otros efectos secundarios desagradables.

El tipo *mensaje* permite que un mensaje esté encapsulado por completo dentro de otro. Este esquema es útil para reenviar, por ejemplo, correo electrónico. Cuando se encapsula un mensaje RFC 822 completo en un mensaje exterior, debe usarse el subtipo *rfc822.*

El subtipo *parcial* hace posible dividir un mensaje encapsulado en pedazos y enviarlos por separado (por ejemplo, si el mensaje encapsulado es demasiado grande). Los parámetros hacen posible reensamblar en forma correcta todas las partes en el destino en el orden correcto.

Por último, el subtipo *externo* puede usarse para mensajes muy grandes (por ejemplo, películas en vídeo). En lugar de incluir el archivo MPEG en el mensaje, se da una dirección FTP y el agente de usuario del receptor puede obtenerlo a través de la red al momento que se requiera. Esta característica es especialmente útil cuando se envía una película a una lista de correo, y sólo se espera que unos cuantos la vean (imagine correo electrónico chatarra que contenga vídeos de propaganda).

El último tipo es *multiparte*, que permite que un mensaje contenga más de una parte, con el comienzo y el fin de cada parte claramente delimitados. El subtipo *mezclado* permite que cada parte sea diferente, sin ninguna estructura adicional impuesta. Muchos programas de correo electrónico permiten que el usuario agregue uno o más archivos adjuntos a un mensaje de texto. Estos archivos adjuntos se envían mediante el tipo *multiparte*.

A diferencia del tipo *multiparte*, el subtipo *alternativa* permite que el mismo mensaje se incluya varias veces, pero expresado en dos o más medios diferentes. Por ejemplo, un mensaje puede enviarse como ASCII común, como texto enriquecido (*enriched*) y como PostScript. Un agente de usuario adecuadamente diseñado que reciba uno de tales mensajes lo presentará en PostScript de ser posible. La segunda posibilidad sería como texto enriquecido. Si ninguna de las dos fuera posible, se presentaría el texto ASCII normal. Las partes deben ordenarse de la más sencilla a la más compleja para ayudar a que los receptores con agentes de usuario pre-MIME puedan encontrarle algún sentido al mensaje (por ejemplo, incluso un usuario pre-MIME puede leer texto ASCII común).

El subtipo *alternativa* también puede servir para varios lenguajes. En este contexto, puede pensarse en la piedra Rosetta como uno de los primeros mensajes *multiparte/alternativa.*

En la figura 7-13 se muestra un ejemplo multimedia. Aquí se transmiten una felicitación de cumpleaños como texto y como canción. Si el receptor tiene capacidad de audio, el agente de usuario traerá el archivo de sonido, *birthday.snd,* y lo ejecutará. Si no, se presenta la letra en la pantalla en absoluto silencio. Las partes están delimitadas por dos guiones seguidos de una cadena (generada por software) especificada en el parámetro *boundary.*

```
From: elinor@abcd.com
To: carolyn@xyz.com
MIME-Version: 1.0
Message-Id: <0704760941.AA00747@abcd.com>
Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
Subject: La Tierra da vuelta al Sol un número entero de veces

Éste es el preámbulo. El agente de usuario lo ignora. Tenga un bonito día.

--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/enriched

Feliz cumpleaños a ti
Feliz cumpleaños a ti
Feliz cumpleaños, querida <bold> Carolyn </bold>
Feliz cumpleaños a ti

--qwertyuiopasdfghjklzxcvbnm
Content-Type: message/external-body;
      access-type="anon-ftp";
      site="bicycle.abcd.com";
      directory="pub";
      name="birthday.snd"

content-type: audio/basic
content-transfer-encoding: base64
--qwertyuiopasdfghjklzxcvbnm--
```

Figura 7-13. Mensaje multiparte que contiene alternativas de texto enriquecido y audio.

Observe que el encabezado *Content-Type* aparece en tres lugares en este ejemplo. En el nivel superior, el encabezado indica que el mensaje tiene varias partes. En cada parte, el encabezado indica el tipo y el subtipo de la parte. Por último, en el texto de la segunda parte, el encabezado es necesario para indicarle al agente de usuario la clase de archivo externo que debe conseguir. Para indicar esta pequeña diferencia de uso, hemos usado letras en minúscula, aunque los encabezados no hacen distinciones entre mayúsculas y minúsculas. De la misma manera, se requiere el encabezado *content-transfer-encoding* para cualquier cuerpo externo que no esté codificado como ASCII de 7 bits.

Regresando a los subtipos para mensajes multiparte, existen dos posibilidades más. El subtipo *paralelo* se usa cuando todas las partes deben "verse" de manera simultánea. Por ejemplo, las películas con frecuencia tienen un canal de audio y uno de vídeo. Las películas son más efectivas si estos dos canales se producen en paralelo, en lugar de consecutivamente.

Por último, el subtipo *compendio* se usa cuando se empacan muchos mensajes juntos en un mensaje compuesto. Por ejemplo, algunos grupos de discusión de Internet recolectan mensajes de los subscriptores y luego los envían como un solo mensaje *multiparte/compendio*.

7.2.4 Transferencia de mensajes

El sistema de transferencia de mensajes se ocupa de transmitir del remitente al destinatario. La manera más sencilla de hacer esto es establecer una conexión de transporte de la máquina de origen a la de destino y sencillamente transferir el mensaje. Después de examinar la manera en que se hace normalmente esto, estudiaremos algunas situaciones en las que no funciona esto, y lo que puede hacerse al respecto.

SMTP—Protocolo Simple de Transporte de Correo

En Internet, el correo electrónico se entrega al hacer que la máquina de origen establezca una conexión TCP con el puerto 25 de la máquina de destino. Escuchando en este puerto está un demonio de correo electrónico que habla con el **SMTP** (**Protocolo Simple de Transporte de Correo**). Este demonio acepta conexiones de entrada y copia mensajes de ellas a los buzones adecuados. Si no puede entregarse un mensaje, se devuelve al remitente un informe de error que contiene la primera parte del mensaje que no pudo entregarse.

SMTP es un protocolo ASCII sencillo. Después de establecer la conexión TCP con el puerto 25, la máquina emisora, operando como cliente, espera que la máquina receptora, operando como servidor, hable primero. El servidor comienza enviando una línea de texto que proporciona su identidad e indica si está preparado o no para recibir correo. Si no lo está, el cliente libera la conexión y lo intenta después.

Si el servidor está dispuesto a aceptar correo electrónico, el cliente anuncia de quién proviene el mensaje, y a quién está dirigido. Si existe el destinatario en el destino, el servidor da al cliente permiso para enviar el mensaje. A continuación el cliente envía el mensaje y el servidor confirma su recepción. Por lo general, no se requieren sumas de verificación porque TCP proporciona un flujo de bytes confiable. Si hay más correo electrónico, se envía ahora. Una vez que todo el correo electrónico ha sido intercambiado en ambas direcciones, se libera la conexión. En la

figura 7-14 se muestra un diálogo de ejemplo para el envío del mensaje de la figura 7-13, incluidos los códigos numéricos usados por SMTP. Las líneas enviadas por el cliente se marcan con *C:*; aquellas enviadas por el servidor se marcan con *S:*.

Pueden ser útiles algunos comentarios sobre la figura 7-14. El primer comando del cliente es ciertamente *HELO.* De las dos abreviaturas de cuatro caracteres de *HELLO,* ésta tiene numerosas ventajas sobre su competidora. La razón por la que los comandos tienen que ser de cuatro caracteres se ha perdido en las brumas del tiempo.

En la figura 7-14 el mensaje se envía a un solo destinatario, por lo que se usa un solo comando *RCPT*. Se permiten tales comandos para enviar un solo mensaje a destinatarios múltiples; se confirma la recepción de cada uno o se rechaza de manera individual. Incluso si se rechazan algunos destinatarios (porque no existen en el destino), el mensaje puede enviarse a los demás.

Por último, aunque la sintaxis de los comandos de cuatro caracteres del cliente se especifica con rigidez, la sintaxis de las respuestas es menos rígida. Sólo cuenta el código numérico. Cada implementación puede poner la cadena que desee después del código.

Para obtener una mejor idea de cómo funcionan SMTP y algunos otros protocolos descritos en este capítulo, pruébelos. En todos los casos, primero vaya a una máquina conectada a Internet. En un sistema UNIX, en una línea de comando, escriba:

```
telnet mail.isp.com 25
```

y sustituya el nombre DNS de su servidor de correo ISP por *mail.isp.com*. En un sistema Windows, haga clic en Inicio | Ejecutar (Start | Run) y después escriba el comando en el cuadro de diálogo. Este comando establecerá una conexión telnet (es decir, TCP) con el puerto 25 de esa máquina. El puerto 25 es el puerto SMTP (vea la figura 6-27 en la que se listan algunos puertos comunes). Probablemente obtendrá una respuesta parecida a lo siguiente:

```
Trying 192.30.200.66...
Connected to mail.isp.com
Escape character is ’ˆ]’.
220 mail.isp.com Smail #74 ready at Thu, 25 Sept 2002 13:26 +0200
```

Las primeras tres líneas son de telnet y le indican lo que están haciendo. La última línea es del servidor SMTP de la máquina remota y anuncia su disponibilidad de hablar con usted y aceptar correo electrónico. Para ver cuáles comandos acepta, escriba

```
HELP
```

De aquí en adelante, es posible una secuencia de comandos como la que se muestra en la figura 7-14, a partir del comando *HELO* del cliente.

Vale la pena mencionar que no es un accidente el uso de líneas de texto ASCII para los comandos. La mayoría de los protocolos de Internet funcionan de esta manera. El uso de texto ASCII hace que los protocolos sean fáciles de probar y depurar. Pueden probarse emitiendo comandos de manera manual, como vimos anteriormente, y las copias de los datos a pantalla o impresora son fáciles de leer.

```
              S: 220 servicio SMTP xyz.com listo
C: HELO abcd.com
              S: 250 xyz.com dice hola a abcd.com
C: MAIL FROM:<elinor@abcd.com>
              S: 250 emisor ok
C: RCPT TO:<carolyn@xyz.com>
              S: 250 receptor ok
C: DATA
              S: 354 envía correo; termina con una línea únicamente con "."
C: From: elinor@abcd.com
C: To: carolyn@xyz.com
C: MIME-Version: 1.0
C: Message-Id: <0704760941.AA00747@abcd.com>
C: Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
C: Subject: La Tierra da vuelta al Sol un número entero de veces
C:
C: Éste es el preámbulo. El agente de usuario lo ignora. Tenga un bonito día.
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: text/enriched
C:
C: Feliz cumpleaños a ti
C: Feliz cumpleaños a ti
C: Feliz cumpleaños, querida <bold> Carolyn </bold>
C: Feliz cumpleaños a ti
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: message/external-body;
C:            access-type="anon-ftp";
C:            site="bicycle.abcd.com";
C:            directory="pub";
C:            name="birthday.snd"
C:
C: content-type: audio/basic
C: content-transfer-encoding: base64
C: --qwertyuiopasdfghjklzxcvbnm
C: .
              S: 250 mensaje aceptado
C: QUIT
              S: 221 xyz.com cerrando conexión
```

Figura 7-14. Transferencia de un mensaje de *elinor@abc.com* a *carolyn@xyz.com*.

Aunque el protocolo SMTP está bien definido, pueden surgir algunos problemas. Uno se relaciona con la longitud del mensaje. Algunas implementaciones más viejas no pueden manejar mensajes mayores que 64 KB. Otro problema se relaciona con las terminaciones de temporizador. Si el cliente y el servidor tienen temporizadores diferentes, uno de ellos puede terminar mientras

que el otro continúa trabajando, terminando en forma inesperada la conexión. Por último, en contadas ocasiones, pueden dispararse tormentas de correo infinitas. Por ejemplo, si el *host* 1 contiene la lista de correo *A* y el *host* 2 contiene la lista de correo *B* y cada lista contiene una entrada para la otra lista, entonces un mensaje enviado a cualquiera de las listas generará una cantidad sin fin de tráfico de correo electrónico, a menos que alguien lo verifique.

Para superar algunos de estos problemas, se ha definido el SMTP extendido (**ESMTP**) en el RFC 2821. Los clientes que deseen usarlo deben enviar inicialmente un mensaje *EHLO*, en lugar de *HELO*. Si el saludo se rechaza, esto indica que el servidor es un servidor SMTP normal, y el cliente debe proceder de la manera normal. Si se acepta el *EHLO*, entonces se permiten los comandos y parámetros nuevos.

7.2.5 Entrega final

Hasta ahora hemos supuesto que todos los usuarios trabajan en máquinas capaces de enviar y recibir correo electrónico. Como vimos, el correo electrónico se entrega al hacer que el emisor establezca una conexión TCP con el receptor y después que envíe el correo electrónico a través de ella. Este modelo funcionó bien por décadas cuando todos los *hosts* ARPANET (y más tarde Internet) se pusieron, de hecho, en línea todo el tiempo para aceptar conexiones TCP.

Sin embargo, con el advenimiento de personas que acceden a Internet llamando a su ISP por medio de un módem, ese modelo dejó de usarse. El problema es el siguiente: ¿qué sucede cuando Elena desea enviar correo electrónico a Carolina y esta última no está en línea en ese momento? Elena no puede establecer una conexión TCP con Carolina y, por lo tanto, no puede ejecutar el protocolo SMTP.

Una solución es que un agente de transferencia de mensajes en una máquina ISP acepte correo electrónico para sus clientes y lo almacene en sus buzones en una máquina ISP. Puesto que este agente puede estar en línea todo el tiempo, el correo electrónico puede enviarse las 24 horas del día.

POP3

Desgraciadamente, esta solución genera otro problema: ¿cómo obtiene el usuario el correo electrónico del agente de transferencia de mensajes del ISP? La solución a este problema es crear otro protocolo que permita que los agentes de transferencia de usuarios (en PCs cliente) contacten al agente de transferencia de mensajes (en la máquina del ISP) y que el correo electrónico se copie desde el ISP al usuario. Tal protocolo es **POP3** (**Protocolo de Oficina de Correos Versión 3**), que se describe en el RFC 1939.

En la figura 7-15(a) se presenta la situación más común (en la que tanto el emisor como el receptor tienen una conexión permanente a Internet). En la figura 7-15(b) se ilustra la situación en la que el emisor está (actualmente) en línea pero el receptor no.

POP3 inicia cuando el usuario arranca el lector de correo. Éste llama al ISP (a menos que ya haya una conexión) y establece una conexión TCP con el agente de transferencia de mensajes en

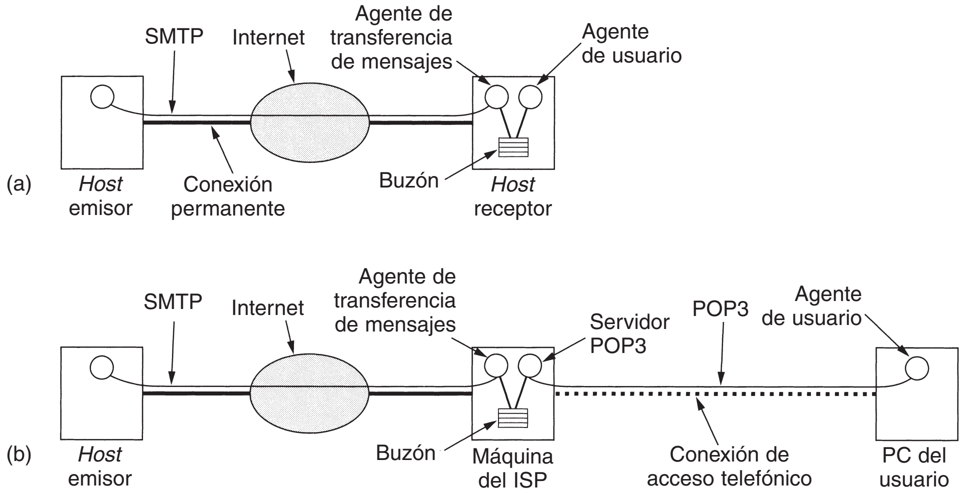


Figura 7-15. (a) Envío y recepción de correo cuando el receptor tiene una conexión permanente a Internet y el agente de usuario se ejecuta en la misma máquina que el agente de transferencia de mensajes. (b) Lectura de correo cuando el receptor tiene una conexión de acceso telefónico a un ISP.

el puerto 110. Una vez que se ha establecido la conexión, el protocolo POP3 pasa por tres estados en secuencia:

1. Autorización.
2. Transacciones.
3. Actualización.

El estado de autorización tiene que ver con el inicio de sesión por parte del usuario. El estado de transacción se relaciona con el hecho de que el usuario colecte los mensajes de correo electrónico y los marque para eliminación desde el buzón. El estado de actualización se encarga de que los mensajes de correo electrónico se eliminen realmente.

Este comportamiento se puede observar tecleando algo como lo siguiente:

```
telnet mail.isp.com 110
```

donde *mail.isp.com* representa el nombre DNS del servidor de correo de su ISP. Telnet establece una conexión TCP con el puerto 110, en el que escucha el servidor POP3. Al aceptar la conexión TCP, el servidor envía un mensaje ASCII que indica que está presente. Por lo general, este mensaje comienza con +OK seguido de un comentario. En la figura 7-16 se muestra un escenario de ejemplo, comenzando después de que se ha establecido la conexión TCP. Al igual que antes, las líneas marcadas con *C:* provienen del cliente (usuario) y las marcadas con *S:* provienen del servidor (agente de transferencia de mensajes en la máquina del ISP).

```
               S: +OK servidor POP3 listo
C: USER carolina
               S: +OK
C: PASS vegetales
               S: +OK inicio de sesión exitoso

C: LIST
               S: 1 2505
               S: 2 14302
               S: 3 8122
               S: .
C: RETR 1
               S: (envía mensaje 1)
C: DELE 1
C: RETR 2
               S: (envía mensaje 2)

C: DELE 2
C: RETR 3
               S: (envía mensaje 3)
C: DELE 3
C: QUIT
               S: +OK servidor POP3 desconectándose
```

Figura 7-16. Uso de POP3 para obtener tres mensajes.

Durante el estado de autorización, el cliente envía su nombre de usuario y después su contraseña. Después de un inicio de sesión exitoso, el cliente puede enviar el comando *LIST*, que causa que el servidor liste el contenido del buzón, un mensaje por línea, y que dé la longitud de ese mensaje. La lista se termina mediante un punto.

A continuación el cliente puede recuperar los mensajes utilizando el comando *RETR* y los puede marcar para eliminarlos con *DELE*. Cuando todos los mensajes se han recuperado (y posiblemente se han marcado para eliminación), el cliente emite el comando *QUIT* para terminar el estado de transacción y entrar al de actualización. Cuando el servidor ha eliminado todos los mensajes, envía una respuesta y termina la conexión TCP.

Si bien es cierto que el protocolo POP3 soporta la capacidad de descargar un mensaje específico o un conjunto de mensajes y dejarlos en el servidor, la mayoría de los programas de correo electrónico simplemente descarga todo y vacía el buzón. Este comportamiento significa que en la práctica, la única copia está en el disco duro del usuario. Si se cae, es posible que se pierda de manera permanente todo el correo electrónico.

Resumamos brevemente la forma en que funciona el correo electrónico para los clientes ISP. Elena utiliza un programa de correo electrónico (es decir, un agente de usuario) para crear un mensaje para Carolina y hace clic en un icono para enviarlo. El programa de correo electrónico entrega el mensaje al agente de transferencia de mensajes del *host* de Elena. Dicho agente ve que el mensaje está dirigido a *carolyn@xyz.com* por lo que utiliza DNS para buscar el registro *MX* de *xyz.com* (donde *xyz.com* es el ISP de Carolina). Esta consulta regresa el nombre del DNS del servidor de correo de *xyz.com*. El agente de transferencia de mensajes ahora busca la dirección IP de

esta máquina utilizando nuevamente DNS, por ejemplo, *gethostbyname*. Después establece una conexión TCP con el servidor SMTP en el puerto 25 de esta máquina. A continuación utiliza una secuencia de comandos SMTP análoga a la de la figura 7-14 para transferir el mensaje al buzón de Carolina y termina la conexión TCP.

A su debido tiempo, Carolina arranca su PC, se conecta a su ISP e inicia su programa de correo electrónico. Éste establece una conexión TCP con el servidor POP3 en el puerto 110 de la máquina servidora de correo del ISP. Por lo general, el nombre del DNS o la dirección IP de esta máquina se configura cuando se instala el programa de correo electrónico o cuando se realiza la suscripción al ISP. Una vez que se ha establecido la conexión TCP, el programa de correo electrónico de Carolina se ejecuta en el protocolo POP3 para obtener el contenido del buzón a su disco duro utilizando comandos similares a los de la figura 7-16. Una vez que se ha transferido todo el correo electrónico, se libera la conexión TCP. De hecho, la conexión con el ISP también puede terminarse ahora, puesto que todo el correo electrónico se encuentra en el disco duro de la máquina de Carolina. Por supuesto, para enviar una respuesta, será necesaria otra vez la conexión con el ISP, por lo que generalmente dicha conexión no se termina después de obtener el correo electrónico.

IMAP

Para un usuario que tiene una cuenta de correo electrónico con un ISP que siempre se accede desde una PC, POP3 es adecuado y se utiliza ampliamente debido a su sencillez y robustez. Sin embargo, es un axioma de la industria de la computación el hecho de que siempre que algo funciona bien, alguien comienza a pedir más características (y a obtener más errores). Eso también sucedió con el correo electrónico. Por ejemplo, muchas personas tienen una sola cuenta de correo electrónico en el trabajo o en la escuela y desean accederla desde el trabajo, desde la PC de su casa, desde su computadora portátil cuando están en viaje de negocios y desde algún cibercafé cuando se encuentran de vacaciones. Aunque POP3 permite esto, debido a que descarga todos los mensajes almacenados en cada contacto, el resultado es que los mensajes de correo electrónico del usuario quedan esparcidos rápidamente en múltiples máquinas, más o menos de manera aleatoria, y algunos de ellos ni siquiera en la máquina del usuario.

Esta desventaja dio lugar a un protocolo de entrega final alternativo, **IMAP** (**Protocolo de Acceso a Mensajes de Internet),** que se define en el RFC 2060. A diferencia de POP3, que asume básicamente que el usuario vaciará el buzón de cada contacto y trabajará sin conexión después de eso, IMAP supone que todo el correo electrónico permanecerá en el servidor de manera indefinida en múltiples buzones de correo. IMAP proporciona mecanismos de gran alcance para leer mensajes o incluso partes de un mensaje, una característica útil cuando se utiliza un módem lento para leer parte del texto de un mensaje dividido en varios fragmentos y que tiene archivos adjuntos grandes de audio y vídeo. Debido a que la suposición más razonable es que los mensajes no se transferirán a la computadora del usuario para un almacenamiento permanente, IMAP proporciona mecanismos para crear, destruir y manipular múltiples buzones en el servidor. De esta forma, un usuario puede mantener un buzón para cada uno de sus contactos y colocar ahí mensajes de la bandeja de entrada después de que se han leído.

IMAP tiene muchas características, como la capacidad de dirigir correo no por número de llegada, como se hace en la figura 7-8, sino utilizando atributos (por ejemplo, dame el primer mensaje

de Juan). A diferencia de POP3, IMAP también puede aceptar correo saliente para enviarlo al destino, así como entregar correo electrónico entrante.

El estilo general del protocolo IMAP es similar al de POP3, como se muestra en la figura 7-16, excepto que hay docenas de comandos. El servidor IMAP escucha en el puerto 143. En la figura 7-17 se muestra una comparación de POP3 e IMAP. Sin embargo, se debe mencionar que no todos los ISPs ni todos los programas de correo electrónico soportan ambos protocolos. Por lo tanto, cuando se elige un programa de correo electrónico, es importante averiguar qué protocolos soporta y asegurarse de que el ISP soporte por lo menos uno de ellos.

Característica	POP3	IMAP
En dónde se define el protocolo	RFC 1939	RFC 2060
Puerto TCP utilizado	110	143
En dónde se almacena el correo electrónico	PC del usuario	Servidor
En dónde se lee el correo electrónico	Sin conexión	En línea
Tiempo de conexión requerido	Poco	Mucho
Uso de recursos del servidor	Mínimo	Amplio
Múltiples buzones	No	Sí
Quién respalda los buzones	Usuario	ISP
Bueno para los usuarios móviles	No	Sí
Control del usuario sobre la descarga	Poco	Mucho
Descargas parciales de mensajes	No	Sí
¿Es un problema el espacio en disco?	No	Con el tiempo podría serlo
Sencillo de implementar	Sí	No
Soporte amplio	Sí	En crecimiento

Figura 7-17. Comparación entre POP3 e IMAP.

Características de entrega

Independientemente de si se utiliza POP3 o IMAP, muchos sistemas proporcionan ganchos para procesamiento adicional de correo electrónico entrante. Una característica especialmente valiosa para muchos usuarios de correo electrónico es la capacidad de establecer **filtros.** Éstas son reglas que se verifican cuando llega el correo electrónico o cuando se inicia el agente de usuario. Cada regla especifica una condición y una acción. Por ejemplo, una regla podría decir que cualquier mensaje del jefe va al buzón número 1, cualquier mensaje de un grupo seleccionado de amigos va al buzón número 2, y cualquier mensaje que contenga en la línea Asunto ciertas palabras objetables se descarta sin ningún comentario.

Algunos ISPs proporcionan un filtro que clasifica de manera automática el correo electrónico como importante o como publicidad no deseada (correo basura) y almacena cada mensaje en el buzón correspondiente. Por lo general, tales filtros funcionan verificando primero si el origen es un *spammer* (persona que envía correo basura) conocido. A continuación, por lo general examinan la

línea de asunto. Si cientos de usuarios han recibido un mensaje con la misma línea de asunto, probablemente sea correo basura. También se utilizan otras técnicas para la detección de correo basura.

Otra característica de entrega que por lo general se proporciona es la capacidad de reenviar (de manera temporal) correo electrónico entrante a una dirección diferente, la cual puede ser una computadora manejada por un servicio de localización comercial, el cual a continuación localiza al usuario por radio o satélite y le indica que tiene un mensaje en su localizador.

Otra característica común de la entrega final es la capacidad de instalar un **demonio de vacaciones.** Éste es un programa que examina cada mensaje entrante y envía al emisor una respuesta grabada como

Hola. Estoy de vacaciones. Regresaré el 24 de agosto. Tenga un buen fin de semana.

Tales respuestas también pueden especificar cómo manejar asuntos urgentes en el ínterin, las personas a quienes se puede acudir en caso de problemas específicos, etcétera. La mayoría de los demonios de vacaciones lleva un registro de a quién le ha enviado respuestas grabadas y se abstiene de enviar a la misma persona una segunda respuesta. Los buenos demonios también verifican si los mensajes entrantes se enviaron a una lista de correo, y de ser así, no envían una respuesta grabada. (Las personas que envían mensajes a listas de correo grandes durante el verano probablemente no desean obtener cientos de respuestas que detallen los planes de vacaciones de todos los demás.)

El autor una vez se topó con una forma extrema de procesamiento de correo cuando envió un mensaje de correo electrónico a una persona que se jactaba de recibir 600 mensajes al día. Su identidad no se revelará aquí, por miedo a que la mitad de los lectores de este libro también le envíen a él correo electrónico. Llamémoslo Juan.

Juan ha instalado un robot de correo electrónico que verifica cada mensaje entrante para ver si proviene de un nuevo remitente. De ser así, regresa una respuesta grabada que explica que Juan ya no puede leer personalmente todo su correo electrónico. En su lugar, ha producido un documento de FAQs (preguntas más frecuentes) personales en el que responde muchas preguntas que generalmente se le formulan. Es común que los grupos de noticias tengan FAQs, no personas.

La FAQ de Juan da su dirección, fax y números telefónicos e indica cómo contactar a su compañía. Explica cómo contratarlo como orador y describe en dónde obtener sus ponencias y otros documentos. También proporciona apuntadores a software que ha escrito, a conferencias que está impartiendo, a un estándar del que es editor, etcétera. Tal vez esta estrategia sea necesaria, pero podría ser que una FAQ personal sea el último símbolo de *status*.

Correo de Web

Un tema final que vale la pena mencionar es el correo de Web. Algunos sitios Web, como Hotmail y Yahoo, proporcionan servicio de correo electrónico a cualquiera que lo desee. Funcionan como se menciona a continuación. Tienen agentes de transferencia de mensajes normales que escuchan el puerto 25 para conexiones SMTP entrantes. Para contactar, digamos, Hotmail, usted tiene que adquirir su registro *MX* de DNS, por ejemplo, tecleando

```
host –a –v hotmail.com
```

en un sistema UNIX. Suponga que el servidor de correo se llama *mx10.hotmail.com*, por lo que al escribir

```
telnet mx10.hotmail.com 25
```

puede establecer una conexión TCP a través de la cual pueden enviarse comandos SMTP de la forma usual. Hasta ahora, nada es inusual, excepto que estos servidores enormes por lo general están ocupados, de modo que tal vez se necesiten varios intentos para que se acepte una conexión TCP.

La parte interesante es la forma en que se entrega el correo electrónico. Básicamente, cuando el usuario va a la página Web de correo electrónico, se despliega un formulario en el que se pide al usuario que introduzca un nombre de usuario y una contraseña. Cuando el usuario hace clic en Registrarse, el nombre de usuario y la contraseña se envían al servidor, quien los valida. Si el inicio de sesión es exitoso, el servidor encuentra el buzón del usuario y construye una lista similar a la de la figura 7-8, sólo que formateada como página Web en HTML. Esta página Web se envía al navegador para su despliegue. En muchos de los elementos de la página se puede hacer clic, por lo que los mensajes se pueden leer, eliminar, etcétera.

7.3 WORLD WIDE WEB

World Wide Web es un armazón arquitectónico para acceder a documentos vinculados distribuidos en miles de máquinas de toda Internet; en diez años, pasó de ser una manera de distribuir datos sobre física de alta energía a la aplicación que millones de personas piensan que es "Internet". Su enorme popularidad se deriva del hecho de que tiene una interfaz gráfica atractiva que es fácil de usar por los principiantes y proporciona un enorme cúmulo de información sobre casi cualquier tema concebible, desde aborígenes hasta zoología.

Web (también conocida como **WWW**) comenzó en 1989 en el CERN, el Centro Europeo de Investigación Nuclear. El CERN tiene varios aceleradores en los que los científicos de los países europeos participantes llevan a cabo investigaciones sobre física de partículas. Estos equipos con frecuencia tienen miembros de media docena de países o más. La mayoría de los experimentos son altamente complejos, y requieren años de planeación adelantada y construcción de equipo. Web surgió de la necesidad de lograr que estos grandes grupos de investigadores dispersos internacionalmente colaboren usando un conjunto siempre cambiante de informes, planos, dibujos, fotos y otros documentos.

La propuesta inicial de una red de documentos vinculados surgió del físico del CERN Tim Berners-Lee en marzo de 1989. El primer prototipo (basado en texto) estaba en operación 18 meses después. En diciembre de 1991 se hizo una demostración pública en la conferencia Hypertext '91 en San Antonio, Texas.

Esta demostración y su publicidad acompañante captó la atención de otros investigadores, lo que llevó a Marc Andreessen de la Universidad de Illinois a comenzar el desarrollo del primer navegador gráfico, Mosaic. Se liberó en febrero de 1993. Mosaic fue tan popular que un año más tarde, Andreessen formó su propia compañía, Netscape Communications Corp., cuya meta era desarrollar clientes, servidores y otro tipo de software Web. Cuando Netscape se liberó en 1995, los

inversionistas, creyendo que éste era el siguiente Microsoft, pagaron 1500 millones de dólares por las acciones. Esta transacción fue muy sorprendente porque la compañía sólo tenía un producto, operaba profundamente en la red y había anunciado que no esperaba obtener utilidades en un futuro previsible. Durante los siguientes tres años, Netscape Navigator y Microsoft Internet Explorer sostuvieron una "guerra de navegadores", cada uno tratando frenéticamente de agregar más características (y, por ende, más errores) que el otro. En 1998, America Online compró Netscape Communications Corp. por 4,200 millones de dólares, terminando con la breve vida de Netscape como compañía independiente.

En 1994, el CERN y el MIT firmaron un acuerdo para establecer el **World Wide Web Consortium** (W3C), una organización dedicada al desarrollo de Web, la estandarización de protocolos y el fomento de interoperabilidad entre los sitios. Berners-Lee se convirtió en el director. Desde entonces, cientos de universidades y compañías se han unido al consorcio. Aunque hay más libros sobre Web de los que pueden contarse, el mejor lugar para recibir información actualizada sobre Web es (naturalmente) la Web misma. La página de inicio del consorcio puede encontrarse en *http://www.w3.org.* Los lectores interesados pueden encontrar ahí vínculos con páginas que cubren todos los documentos y actividades del consorcio.

7.3.1 Panorama de la arquitectura

Desde el punto de vista del usuario, Web consiste en un enorme conjunto de documentos a nivel mundial, generalmente llamados **páginas Web.** Cada página puede contener vínculos (apuntadores) a otras páginas relacionadas en cualquier lugar del mundo. Los usuarios pueden seguir un vínculo haciendo clic en él, lo que los lleva a la página apuntada. Este proceso puede repetirse de manera indefinida. La idea de hacer que una página apunte a otra, lo que ahora se conoce como **hipertexto,** fue inventada por un profesor visionario de ingeniería eléctrica del MIT, Vannevar Bush, en 1945, mucho antes de que se inventara Internet.

Las páginas se ven mediante un programa llamado **navegador;** Internet Explorer y Netscape son dos de los navegadores más populares. El navegador obtiene la página solicitada, interpreta el texto y los comandos de formateo que contienen, y despliega la página, adecuadamente formateada, en la pantalla. En la figura 7-18(a) se da un ejemplo. Al igual que en muchas páginas Web, ésta comienza con un título, contiene cierta información y termina con la dirección de correo electrónico del encargado de mantenimiento de la página. Las cadenas de texto que son vínculos a otras páginas, llamadas **hipervínculos,** se resaltan, ya sea mediante subrayado, presentación en un color especial, o ambas cosas. Para seguir un vínculo, el usuario coloca el cursor en el área resaltada (usando el ratón o las teclas de flecha) y la selecciona (haciendo clic con un botón del ratón u oprimiendo ENTRAR). Aunque existen algunos navegadores no gráficos, como Lynx, no son tan comunes como los navegadores gráficos, por lo que nos concentraremos en estos últimos. También se están desarrollando navegadores basados en voz.

Los usuarios con curiosidad respecto al Departamento de psicología animal pueden aprender más sobre él haciendo clic en su nombre (subrayado). A continuación el navegador trae la página

BIENVENIDOS A LA PÁGINA DE INICIO DE WWW DE LA UNIVERSIDAD DE EAST PODUNK

- Información de la Universidad
  - Información sobre inscripciones
  - Mapa de las instalaciones
  - Indicaciones para llegar a las instalaciones
  - El cuerpo estudiantil de la UEP

- Departamentos académicos
  - Departamento de psicología animal
  - Departamento de estudios alternativos
  - Departamento de cocina microbiótica
  - Departamento de estudios no tradicionales
  - Departamento de estudios tradicionales

Webmaster@eastpodunk.edu

(a)

DEPARTAMENTO DE PSICOLOGÍA ANIMAL

- Información sobre posibles carreras
- Personal
  - Miembros del profesorado
  - Estudiantes de postgrado
  - Personal no académico
- Proyectos de investigación
- Puestos disponibles
- Nuestros cursos más populares
  - Manejo de herbívoros
  - Administración de caballos
  - Negociando con su mascota
  - Construcción de perreras amables con el usuario
- Lista completa de cursos

Webmaster@animalpsyc.eastpodunk.edu

(b)

Figura 7-18. (a) Página Web. (b) Página accedida al hacer clic en Departamento de psicología animal.

a la que está vinculado el nombre y la despliega, como se muestra en la figura 7-18(b). También es posible hacer clic en los elementos subrayados que se muestran en la figura para desplegar otras páginas, y así se puede continuar. La página nueva puede estar en la misma máquina que la primera, o en otra máquina del otro lado del mundo. El usuario no puede saberlo. El navegador es quien realiza la obtención de páginas sin ayuda del usuario. Si éste llega a regresar a la página principal,

los vínculos ya seguidos tal vez aparezcan con un subrayado punteado (y posiblemente con otro color) para distinguirlos de los vínculos que no se han seguido. Observe que hacer clic en la línea de *Información de la Universidad* de la página principal no tiene ningún efecto; no está subrayada, lo que significa que simplemente es texto y no está vinculada con otra página.

En la figura 7-19 se muestra el modelo básico de la forma en que funciona Web. Aquí, el navegador despliega una página Web en la máquina cliente. Cuando el usuario hace clic en una línea de texto que está vinculada a una página del servidor *abcd.com*, el navegador sigue el hipervínculo enviándole un mensaje al servidor *abcd.com* en el que le solicita la página. Cuando ésta llega, se despliega. Si contiene un hipervínculo a una página del servidor *xyz.com* en el que se hace clic, el navegador a continuación envía un mensaje a dicha máquina solicitando esa página, y así de forma indefinida.

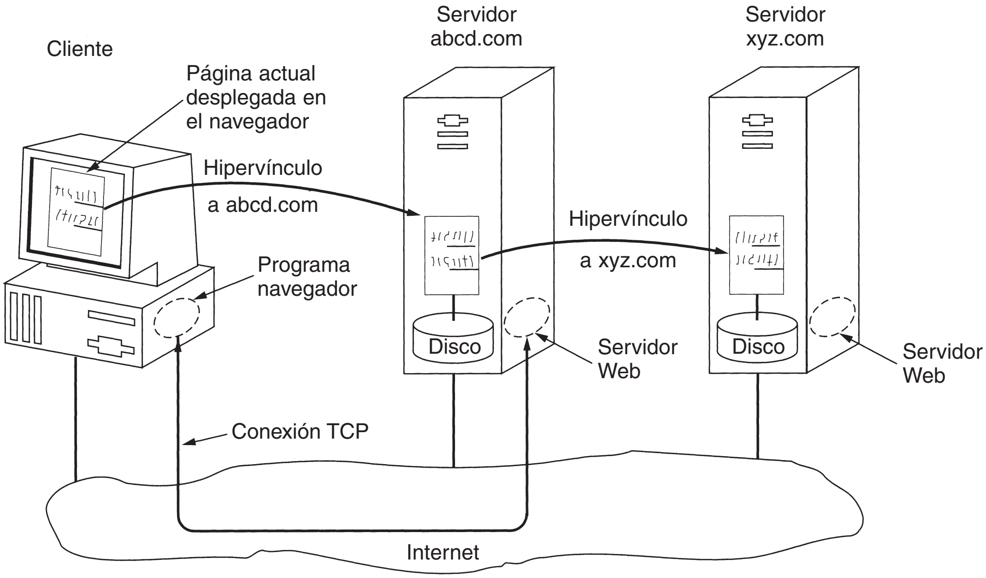


Figura 7-19. Las partes del modelo Web.

El cliente

Ahora examinemos el cliente de la figura 7-19 con mayor detalle. En esencia, un navegador es un programa que puede desplegar una página Web y atrapar los clics que se hacen en los elementos de la página desplegada. Cuando se selecciona un elemento, el navegador sigue el hipervínculo y obtiene la página seleccionada. Por lo tanto, el hipervínculo incrustado necesita una manera de nombrar cualquier página que se encuentre en Web. Las páginas se nombran utilizando **URLs (Localizadores Uniformes de Recursos).** Un URL típico es

http://www.abcd.com/productos.html

Analizaremos los URLs más adelante en este capítulo. Por el momento, es suficiente saber que un URL tiene tres partes: el nombre del protocolo (*http*), el nombre DNS de la máquina donde se localiza la página (*www.abcd.com*) y (por lo general) el nombre del archivo que contiene la página (*productos.html*).

Cuando un usuario hace clic en un hipervínculo, el navegador lleva a cabo una serie de pasos para obtener la página a la que se está apuntado. Suponga que un usuario está navegando Web y encuentra un vínculo sobre telefonía de Internet que apunta a la página de inicio de la ITU, que es *http://www.itu.org/home/index.html*. Listemos los pasos que se dan cuando se selecciona este vínculo.

1. El navegador determina el URL (enviando lo que se seleccionó).
2. El navegador pide al DNS la dirección IP de *www.itu.org*.
3. DNS responde con 156.106.192.32.
4. El navegador realiza una conexión TCP con el puerto 80 en 156.106.192.32.
5. Después envía un mensaje en el que solicita el archivo */home/index.html*.
6. El servidor *www.itu.org* envía el archivo */home/index.html*.
7. Se libera la conexión TCP.
8. El navegador despliega todo el texto de */home/index.html*.
9. El navegador obtiene y despliega todas las imágenes del archivo.

Muchos navegadores despliegan en una línea de estado, que se encuentra en la parte inferior de la pantalla, qué paso están ejecutando actualmente. De esta manera, cuando el desempeño es pobre, el usuario puede ver si se debe a que el DNS o el servidor no están respondiendo, o simplemente hay congestionamiento en la red durante la transmisión de la página.

Para poder desplegar la nueva página (o cualquiera), el navegador tiene que entender su formato. Para permitir que todos los navegadores entiendan todas las páginas Web, éstas se escriben en un lenguaje estandarizado llamado HTML, el cual las describe. Más adelante en este capítulo analizaremos esto con detalle.

Aunque un navegador es básicamente un intérprete HTML, la mayoría de los navegadores tiene varios botones y características para facilitar la navegación en Web. La mayoría tiene un botón para regresar a la página anterior, uno para ir a la siguiente (el cual sólo funciona una vez que el usuario ha regresado de esa página) y uno para ir directamente a la página de inicio del usuario. La mayoría de los navegadores tiene un botón o elemento para establecer un marcador y otro para desplegar la lista de marcadores, lo que hace posible volver a visitar cualquiera de ellos con sólo algunos clics del ratón. Las páginas también pueden guardarse en disco o imprimirse. Por lo general, hay varias opciones disponibles para controlar el diseño de la pantalla y establecer varias preferencias de usuario.

Además de tener texto ordinario (que no está subrayado) e hipertexto (texto subrayado), las páginas Web también pueden tener iconos, dibujos de líneas, mapas y fotografías. Cada uno de

éstos puede (opcionalmente) vincularse a otra página. Hacer clic en alguno de estos elementos causa que el navegador obtenga la página vinculada y la despliegue en pantalla, al igual que al hacer clic en el texto. En el caso de imágenes como fotos y mapas, la página que se obtenga a continuación depende de en cuál parte de la imagen se hizo clic.

No todas las páginas contienen HTML. Una página puede consistir en un documento con formato PDF, un icono con formato GIF, una fotografía con formato JPEG, una canción con formato MP3, un vídeo con formato MPEG, o cualquiera de los cientos de los otros tipos de archivos. Puesto que las páginas HTML estándar pueden vincular cualquiera de éstos, el navegador tiene un problema cuando encuentra una página que no puede interpretar.

En lugar de agrandar cada vez más los navegadores incorporándoles intérpretes para una colección creciente de tipos de archivos, la mayoría de los navegadores ha elegido una solución más general. Cuando un servidor regresa una página, también regresa alguna información adicional acerca de ella. Dicha información incluye el tipo MIME de la página (vea la figura 7-12). Las páginas del tipo *text/html* se despliegan de manera directa, como las páginas de algunos otros tipos integrados. Si el tipo MIME no es de los integrados, el navegador consulta su tabla de tipos MIME que le indique cómo desplegar la página. Esta tabla asocia un tipo MIME con un visor.

Hay dos posibilidades: *plug-ins* y aplicaciones auxiliares. Un ***plug-in*** (conector) es un módulo de código que el navegador obtiene de un directorio especial del disco y lo instala como una extensión de sí mismo, como se ilustra en la figura 7-20(a). Debido a que los *plug-ins* se ejecutan dentro del navegador, tienen acceso a la página actual y pueden modificar su apariencia. Después de que el *plug-in* ha hecho su trabajo (por lo general después de que el usuario se ha movido a una página Web diferente), se elimina de la memoria del navegador.

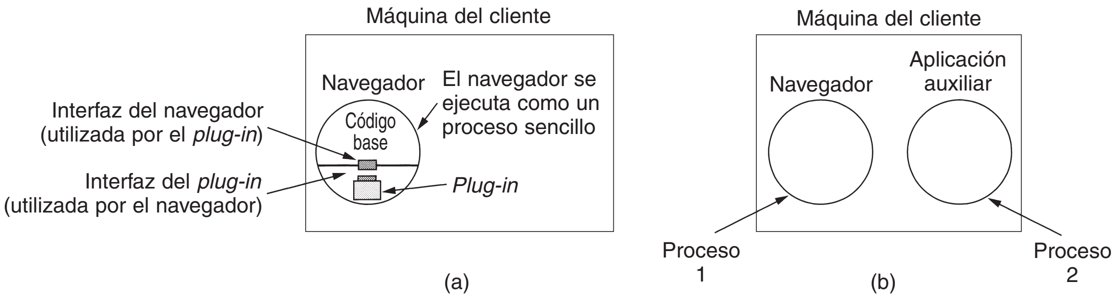


Figura 7-20. (a) Un *plug-in* de navegador. (b) Una aplicación auxiliar.

Cada navegador tiene un conjunto de procedimientos que todos los *plug-ins* tienen que implementar a fin de que el navegador pueda llamarlos. Por ejemplo, generalmente hay un procedimiento que el código base del navegador llama para proporcionar datos que desplegar al *plug-in*. Este conjunto de procedimientos es la interfaz del *plug-in* y es específico del navegador.

Además, pone a disposición del *plug-in* un conjunto de sus propios procedimientos, que proporcionan servicios a los *plug-ins*. Los procedimientos típicos de la interfaz del navegador son para asignar y liberar memoria, desplegar un mensaje en la línea de estado del navegador y consultar al navegador sobre los parámetros.

Para poder utilizar un *plug-in*, primero debe instalarse. El proceso de instalación común consiste en que el usuario vaya al sitio Web del *plug-in* y descargue un archivo de instalación. En Windows, por lo general este archivo es un zip que se extrae en forma automática y que tiene la extensión *.exe*. Cuando se hace doble clic en dicho archivo, se ejecuta un pequeño programa insertado al frente del archivo zip. Este programa extrae el *plug-in* y lo copia en el directorio de *plug-ins* del navegador. A continuación hace las llamadas apropiadas para registrar el tipo MIME del *plug-in* y para asociarlo con éste. En UNIX, el instalador es por lo general una secuencia de comandos (*script*) de shell que maneja la copia y el registro.

La otra forma de ampliar un navegador es utilizar una **aplicación auxiliar.** Ésta es un programa completo que se ejecuta como un proceso independiente [vea la figura 7-20(b)]. Debido a esto no ofrece interfaz con el navegador y no utiliza los servicios de éste. En su lugar, por lo general simplemente acepta el nombre de un archivo de trabajo en el que se ha almacenado el archivo de contenido, abre dicho archivo y despliega su contenido. Por lo general, las aplicaciones auxiliares son programas grandes que existen independientemente del navegador, como Acrobat Reader de Adobe para desplegar archivos PDF o Microsoft Word. Algunos programas (como Acrobat) tienen un *plug-in* que invoca a la aplicación auxiliar misma.

Muchas aplicaciones auxiliares utilizan el tipo MIME *aplicación*. Se ha definido un número considerable de subtipos, por ejemplo, *aplicación/pdf* para los archivos PDF y *aplicación/msword* para archivos de Word. De esta manera, un URL puede apuntar en forma directa a un archivo PDF o Word, y cuando el usuario hace clic en él, se inician automáticamente Acrobat o Word y se les proporciona el mismo nombre de un archivo de trabajo que contiene los datos a desplegar. En consecuencia, es posible configurar a los navegadores para manejar un número virtualmente ilimitado de tipos de documento sin tener que modificar el navegador. Los servidores Web modernos con frecuencia están configurados con cientos de combinaciones de tipos/subtipos y se agregan más cada vez que se instala un nuevo programa.

Las aplicaciones auxiliares no están restringidas a utilizar el tipo MIME *aplicación*. Por ejemplo, Adobe Photoshop utiliza *imagen/x-photoshop* y RealOne Player tiene la capacidad de manejar *audio/mp3*.

En Windows, cuando se instala un programa en la computadora, ésta registra los tipos MIME que necesita manejar. Este mecanismo causa un conflicto cuando múltiples visores están disponibles para algún subtipo, como *video/mpg*. Lo que sucede es que el último programa que se registra sobrescribe las asociaciones existentes (tipo MIME, aplicación auxiliar) con las que requiere para sí mismo. Como consecuencia, instalar un nuevo programa podría cambiar la forma en que un navegador maneja los tipos existentes.

En UNIX, este proceso de registro por lo general no es automático. El usuario tiene que actualizar de manera manual ciertos archivos de configuración. Este método significa más trabajo pero menos sorpresas.

Los navegadores también pueden abrir archivos locales, en lugar de obtenerlos de los servidores Web remotos. Debido a que los archivos locales no tienen tipos MIME, el navegador necesita alguna manera para determinar cuál *plug-in* o aplicación auxiliar utilizar para los tipos que no sean sus tipos integrados, como *texto/html* e *imagen/jpeg*. Para manejar archivos locales, las aplicaciones auxiliares pueden asociarse con una extensión de archivo, así como con un tipo MIME. Con

la configuración estándar, *foo.pdf* se abrirá en Acrobat y *bar.doc* se abrirá en Word. Algunos navegadores utilizan el tipo MIME, la extensión de archivo e incluso información tomada del archivo mismo para adivinar el tipo MIME. En particular, Internet Explorer se apoya más fuertemente en la extensión de archivo que en el tipo MIME, cuando puede.

Aquí también pueden surgir conflictos debido a que muchos programas están dispuestos, de hecho ansiosos, a manejar, digamos, *.mpg*. Durante la instalación, los programas que están diseñados para los profesionales por lo general despliegan casillas de verificación para los tipos MIME y las extensiones que pueden manejar a fin de permitir que el usuario seleccione los apropiados y, de esta forma, no sobrescriba las asociaciones existentes por accidente. Los programas dirigidos al mercado consumidor asumen que el usuario no tiene idea de lo que es un tipo MIME y simplemente toman todos los que pueden sin importarles lo que los programas instalados anteriormente hayan hecho.

La capacidad de ampliar el navegador con un número significativo de tipos nuevos es conveniente pero también puede generar problemas. Cuando Internet Explorer obtiene un archivo con la extensión *exe*, sabe que este archivo es un programa ejecutable y, por lo tanto, no tiene una aplicación auxiliar. La acción obvia es ejecutar el programa. Sin embargo, esto podría ser un enorme hoyo de seguridad. Todo lo que un sitio Web malicioso tiene que hacer es producir una página Web con imágenes de, digamos, estrellas de cine o héroes de deportes, todo lo cual se vincula con un virus. Un solo clic en una imagen podría causar la obtención de un programa ejecutable potencialmente hostil, y su ejecución en la máquina del usuario. Para evitar huéspedes no deseados como éstos, es posible configurar Internet Explorer para que sea selectivo al ejecutar automáticamente programas desconocidos, pero no todos los usuarios saben cómo manejar la configuración.

En UNIX puede ocurrir un problema análogo con las secuencias de comandos de shell, pero eso requiere que el usuario instale de manera consciente el shell como una aplicación auxiliar. Por fortuna, esta instalación es tan complicada que nadie podría realizarla por accidente (y pocas personas pueden hacerlo de manera intencional).

El servidor

Basta de hablar sobre el cliente. Ahora echemos un vistazo al servidor. Como vimos anteriormente, cuando el usuario teclea un URL o hace clic en una línea de hipertexto, el navegador lo analiza e interpreta la parte entre *http://* y la siguiente diagonal como un nombre DNS a buscar. Una vez que el navegador tiene la dirección IP del servidor, establece una conexión TCP con el puerto 80 de ese servidor. A continuación envía un comando que contiene el resto del URL, que es el nombre del archivo que se encuentra en ese servidor. Éste regresa el archivo para que el navegador lo despliegue.

Como una primera aproximación, un servidor Web es similar al de la figura 6-6. A éste, al igual que a un servidor Web real, se le proporciona el nombre del archivo a buscar y regresar. En ambos casos, los pasos que da el servidor en su ciclo principal son:

1. Acepta una conexión TCP de un cliente (un navegador).
2. Obtiene el nombre del archivo solicitado.

3. Obtiene el archivo (del disco).

4. Regresa el archivo al cliente.

5. Libera la conexión TCP.

Los servidores Web actuales tienen más características, pero en esencia, esto es lo que hacen.

El problema de este diseño es que cada solicitud requiere un acceso al disco para obtener el archivo. El resultado es que el servidor Web no puede atender más solicitudes por segundo que accesos al disco. Un disco SCSI de alta calidad tiene un tiempo de acceso promedio de aproximadamente 5 mseg, lo que limita al servidor a lo mucho 200 solicitudes/seg, o menos si se tienen que leer con frecuencia archivos grandes. Para un sitio Web grande, esta cifra es muy baja.

Una mejora obvia (utilizada por todos los servidores Web) es mantener un caché en la memoria de los *n* archivos más recientemente utilizados. Antes de ir al disco para obtener un archivo, el servidor verifica el caché. Si el archivo está ahí, se puede proporcionar directamente desde memoria, con lo que se elimina el acceso al disco. Aunque el almacenamiento en caché efectivo requiere una gran cantidad de memoria principal y tiempo de procesamiento extra para verificar el caché y manejar su contenido, el ahorro de tiempo justifica la sobrecarga y costo implícitos.

El siguiente paso para construir un servidor más rápido es hacerlo de múltiples subprocesos. En un diseño, el servidor consiste en un módulo de *front end* que acepta todas las solicitudes entrantes y *k* módulos de procesamiento, como se muestra en la figura 7-21. Los $k + 1$ subprocesos pertenecen al mismo proceso por lo que todos los módulos de procesamiento tienen acceso al caché dentro del espacio de direcciones del proceso. Cuando llega una solicitud, el *front end* la acepta y construye un registro corto que la describe. Después entrega el registro a uno de los módulos de procesamiento. En otro diseño posible, se elimina el *front end* y cada módulo de procesamiento trata de adquirir sus propias solicitudes, pero se necesita un protocolo de bloqueo para evitar conflictos.

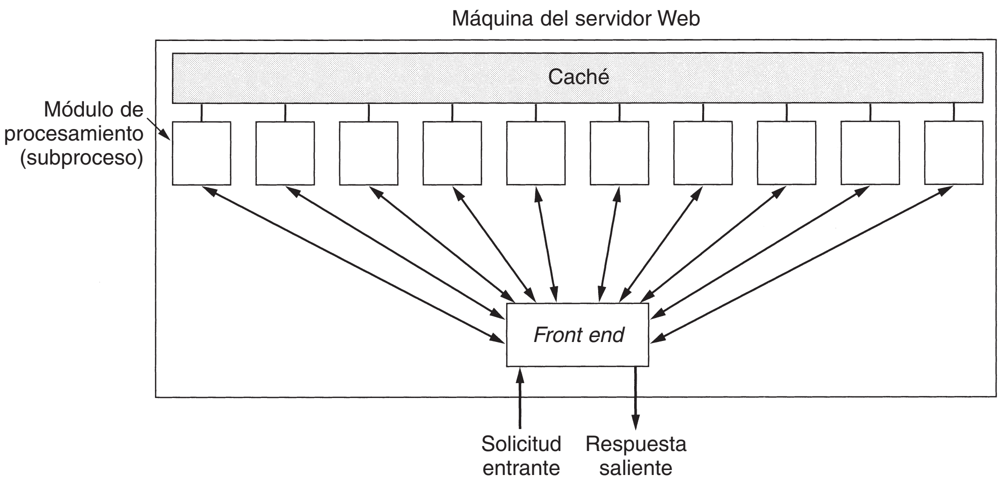


Figura 7-21. Un servidor Web con múltiples subprocesos con un *front end* y módulos de procesamiento.

El módulo de procesamiento primero verifica el caché para ver si el archivo solicitado está ahí. De ser así, actualiza el registro para incluir un apuntador al archivo del registro. Si no está ahí, el módulo de procesamiento inicia una operación de disco para leerlo en el caché (posiblemente descartando algunos otros archivos en caché para hacerle espacio). Cuando el archivo llega del disco, se coloca en el caché y también se regresa al cliente.

La ventaja de este esquema es que mientras que uno o más módulos de procesamiento están bloqueados esperando a que termine una operación del disco (y, por lo tanto, no consumen tiempo de CPU), otros módulos pueden estar trabajando activamente en otras solicitudes. Por supuesto, para obtener cualquier mejora real sobre el modelo de un solo subproceso, es necesario tener múltiples discos, a fin de que más de un disco pueda estar ocupado al mismo tiempo. Con *k* módulos de procesamiento y *k* discos, la velocidad real de transporte puede ser *k* veces mayor que con el servidor de un solo subproceso y un disco.

En teoría, un servidor de un solo subproceso y *k* discos también puede ganar un factor de *k*, pero el código y la administración son mucho más complicados puesto que las llamadas de sistema READ de bloqueo normal no se pueden utilizar para acceder al disco. Con un servidor de múltiples subprocesos se pueden utilizar puesto que READ sólo bloquea el subproceso que hizo la llamada, no todo el proceso.

Los servidores Web modernos hacen más que sólo aceptar y regresar nombres de archivos. De hecho, el procesamiento real de cada solicitud puede ser muy complicado. Por esta razón, en muchos servidores cada módulo de procesamiento realiza una serie de pasos. El *front end* pasa cada solicitud entrante al primer módulo disponible, que después la transporta mediante algunos de los siguientes pasos, dependiendo de los que sean necesarios para esa solicitud en particular.

1. Resuelve el nombre de la página Web solicitada.
2. Autentica al cliente.
3. Realiza control de acceso en el cliente.
4. Realiza control de acceso en la página Web.
5. Verifica el caché.
6. Obtiene del disco la página solicitada.
7. Determina el tipo MIME que se incluirá en la respuesta.
8. Se encarga de diversos detalles.
9. Regresa la respuesta al cliente.
10. Realiza una entrada en el registro del servidor.

El paso 1 es necesario porque la solicitud entrante tal vez no contenga el nombre real del archivo como una cadena literal. Por ejemplo, considere el URL *http://www.cs.vu.nl*, que tiene un nombre de archivo vacío. Tiene que expandirse a algún nombre de archivo predeterminado. Además, los navegadores modernos pueden especificar el lenguaje predeterminado del usuario (por ejemplo,

italiano o inglés), lo cual posibilita que el servidor seleccione una página Web en ese lenguaje, si está disponible. En general, la expansión de nombres no es tan trivial como podría parecer a primera vista, debido a una variedad de convenciones acerca de la asignación de nombres de archivos.

El paso 2 consiste en verificar la identidad del cliente. Este paso es necesario para las páginas que no están disponibles para el público en general. Más adelante en este capítulo analizaremos una forma de hacerlo.

El paso 3 verifica si hay restricciones con respecto a si la solicitud se puede satisfacer a partir de la identidad y ubicación del cliente. El paso 4 verifica si hay restricciones de acceso asociadas con la página misma. Si cierto archivo (por ejemplo, *.htaccess*) se encuentra en el directorio donde se localiza la página deseada, ésta puede prohibir que dominios particulares accedan al archivo; por ejemplo, tal vez sólo permita que usuarios que están dentro de la compañía accedan al archivo.

Los pasos 5 y 6 consisten en obtener la página. El paso 6 necesita poder manejar múltiples lecturas de disco al mismo tiempo.

El paso 7 consiste en determinar el tipo MIME a partir de la extensión del archivo, de las primeras palabras del archivo, de un archivo de configuración y, posiblemente, de otros recursos. El paso 8 tiene que ver con una variedad de tareas, como la construcción de un perfil de usuario o la recolección de ciertas estadísticas.

El paso 9 tiene que ver con el lugar al que se envía el resultado, y el paso 10 crea una entrada en el registro del sistema para propósitos administrativos. Tales registros pueden examinarse en busca de información valiosa acerca del comportamiento de los usuarios, como el orden en que acceden las páginas.

Si llegan demasiadas solicitudes cada segundo, la CPU no será capaz de manejar la carga de procesamiento, sin importar cuántos discos se utilicen en paralelo. La solución es agregar más nodos (computadoras), posiblemente con discos replicados para evitar que los discos se vuelvan el siguiente cuello de botella. Esto lleva al modelo de **granja de servidores** de la figura 7-22. Un *front end* aún acepta solicitudes entrantes pero las distribuye en múltiples CPUs en lugar de en múltiples subprocesos para reducir la carga en cada computadora. Las máquinas individuales podrían contar con múltiples subprocesos y canales como en el esquema anterior.

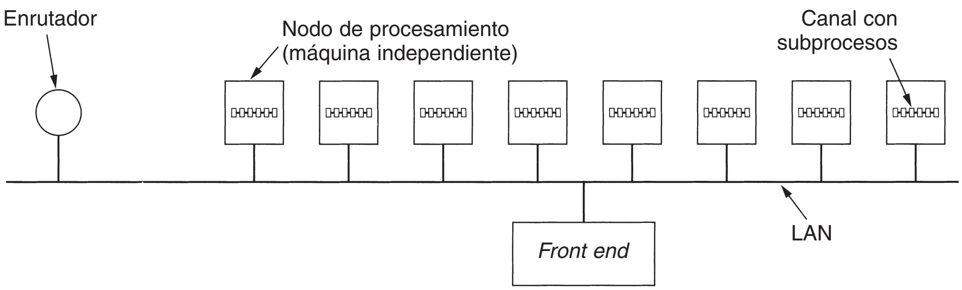


Figura 7-22. Granja de servidores.

Un problema con las granjas de servidores es que no hay caché compartido debido a que cada nodo de procesamiento tiene su propia memoria —a menos que se utilice un multiprocesador

de memoria compartida costoso. Una forma de medir esta pérdida de rendimiento es hacer que un *front end* registre el lugar a donde envía cada respuesta y que envíe las solicitudes subsecuentes de la misma página al mismo nodo. Hacer esto ocasiona que cada nodo sea un especialista en ciertas páginas a fin de que el espacio en caché no se desperdicie al tener cada archivo en cada caché.

Otro problema con las granjas de servidores es que la conexión TCP del cliente termine en el *front end*, por lo que la respuesta puede pasar a través del *front end*. Esta situación se ilustra en la figura 7-23(a), donde la solicitud (1) y la respuesta (4) pasan a través del *front end*. Para solucionar este problema, algunas veces se utiliza un truco llamado **transferencia TCP** (**TCP handoff**)**.** Con ésta, el punto final de la conexión TCP se pasa al nodo de procesamiento a fin de que pueda contestar directamente al cliente, que se muestra como (3) en la figura 7-23(b). Esta transferencia se realiza de tal forma que es transparente para el cliente.

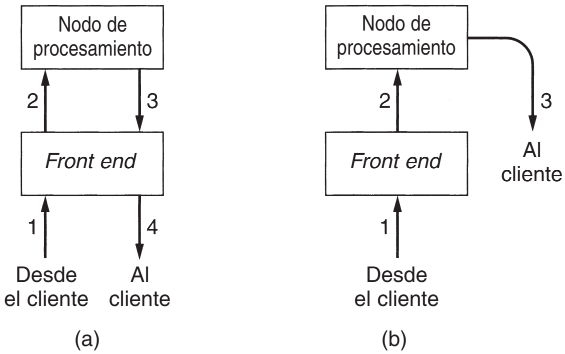


Figura 7-23. (a) Secuencia normal de mensajes solicitud-respuesta. (b) Secuencia cuando se utiliza la transferencia TCP.

URLs—Localizadores Uniformes de Recursos

Hemos dicho repetidamente que las páginas Web pueden contener apuntadores a otras páginas Web. Ahora es tiempo de ver la implementación de estos apuntadores. Cuando se creó Web, de inmediato fue evidente que para que una página Web apuntara a otra se requerían mecanismos para nombrar y localizar las páginas. En particular, había tres preguntas que debían contestarse antes de poder presentar visualmente una página:

1. ¿Cómo se llama la página?
2. ¿Dónde está la página?
3. ¿Cómo se puede acceder a la página?

Si cada página tuviera asignado de alguna manera un nombre único, no habría ambigüedad al identificar las páginas. No obstante, el problema no estaría resuelto. Considere un paralelismo

entre gente y páginas. En Estados Unidos casi todos tienen un número de seguro social, que es un identificador único, puesto que no hay dos personas que tengan el mismo. Sin embargo, armados sólo con el número de seguro social, no hay manera de encontrar la dirección del dueño y, ciertamente, tampoco de saber si se debe escribir a la persona en inglés, español o chino. Web tiene básicamente los mismos problemas.

La solución escogida identifica las páginas de una manera que resuelve los tres problemas a la vez. A cada página se le asigna un **URL** (**Localizador Uniforme de Recursos**) que sirve efectivamente como nombre mundial de la página. Los URLs tienen tres partes: el protocolo (también llamado **esquema),** el nombre DNS de la máquina en donde se encuentra la página y un nombre local que indica de manera única la página específica (por lo general, sólo un nombre de archivo de la máquina en que reside). Por ejemplo, el sitio Web del departamento del autor contiene varios vídeos sobre la universidad y la ciudad de Ámsterdam. El URL de la página de los vídeos es

```
http://www.cs.vu.nl/video/index-en.html
```

Este URL consta de tres partes: el protocolo (*http*), el nombre DNS del *host* (*www.cs.vu.nl*) y el nombre del archivo (*video/index-en.html*), con cierta puntuación que separa las partes. El nombre de archivo es una ruta relativa al directorio Web predeterminado en *cs.vu.nl*.

Muchos sitios cuentan con ciertos atajos para los nombres de archivos. En muchos sitios, un nombre de archivo nulo tiende a ser de manera predeterminada la página de inicio de la organización. Por lo general, cuando el archivo nombrado es un directorio, esto implica un archivo nombrado *index.html*. Por último, *~user/* podría tener una correspondencia con el directorio WWW de *user* y con el archivo *index.html* de ese directorio. Por tanto, la página de inicio del autor puede encontrarse en

```
http://www.cs.vu.nl/~ast/
```

aunque el nombre de archivo real es *index.html* en un directorio predeterminado.

Ahora debe quedar clara la manera en que funciona el hipertexto. Para que pueda hacerse clic en una parte del texto, el redactor de la página debe proporcionar dos elementos de información: el texto en el que se puede hacer clic y el URL de la página a la que se debe ir si se hace clic en dicho texto. Explicaremos la sintaxis del comando más adelante en este capítulo.

Al seleccionarse el texto, el navegador busca el nombre del *host* usando DNS. Armado ahora con la dirección IP del *host,* el navegador establece una conexión TCP con el *host*, y envía por esa conexión el nombre de archivo usando el protocolo especificado. Lotería. La página regresa.

Este esquema de URL es abierto en el sentido de que es directo hacer que los navegadores utilicen múltiples protocolos para obtener diferentes tipos de recursos. De hecho, se han definido los URLs de varios otros protocolos comunes. En la figura 7-24 se listan formas ligeramente simplificadas de los más comunes.

Examinemos con brevedad la lista. El protocolo *http* es el lenguaje nativo de Web, el que hablan los servidores Web. **HTTP** significa **Protocolo de Transferencia de Hipertexto.** Lo examinaremos con mayor detalle, más adelante en este capítulo.

Nombre	Usado para	Ejemplo
http	Hipertexto (HTML)	http://www.cs.vu.nl/~ast/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Archivo local	file:///usr/suzanne/prog.c
news	Grupo de noticias	news:comp.os.minix
news	Artículo	news:AA0134223112@cs.utah.edu
gopher	Gopher	gopher://gopher.tc.umn.edu/11/Libraries
mailto	Envío de correo electrónico	mailto:JohnUser@acm.org
telnet	Inicio de sesión remota	telnet://www.w3.org:80

Figura 7-24. Algunos URLs comunes.

El protocolo *ftp* se usa para acceder a archivos mediante FTP, el protocolo de transferencia de archivos de Internet. FTP tiene más de dos décadas de existencia y está bien atrincherado. Numerosos servidores FTP de todo el mundo permiten que gente de cualquier lugar de Internet establezca una sesión y descargue los archivos colocados en el servidor FTP. Web no cambia esto; simplemente hace más sencilla la obtención de archivos mediante FTP, puesto que FTP tiene una interfaz un tanto arcaica (pero es más poderoso que HTTP; por ejemplo, permite que un usuario en la máquina *A* transfiera un archivo de la máquina *B* a la máquina *C*).

Es posible acceder a un archivo local como página Web, ya sea usando el protocolo *file* o, más sencillamente, con sólo nombrarlo. Este enfoque es parecido a usar FTP pero no requiere un servidor. Por supuesto que sólo funciona con archivos locales.

Mucho antes de que existiera Internet, ya existía el sistema de noticias USENET, el cual consiste en 30,000 grupos de noticias en los que millones de personas discuten una variedad de temas publicando y leyendo artículos relacionados con ese tema. El protocolo de noticias puede utilizarse para llamar a un artículo de noticias como si fuera una página Web. Esto significa que un navegador Web es un lector de noticias. De hecho, muchos navegadores tienen botones o elementos de menú para que la lectura de noticias USENET sea más fácil que utilizar los lectores de noticias estándar.

Se soportan dos formatos para el protocolo *news.* El primero especifica un grupo de noticias y puede usarse para obtener una lista de artículos de un sitio de noticias preconfigurado. El segundo requiere que se indique el identificador de un artículo específico, en este caso *AA0134223112@cs.utah.edu.* A continuación el navegador obtiene el artículo dado de su sitio de noticias preconfigurado usando **NNTP** (**Protocolo de Transferencia de Noticias en Red**)**.** No estudiaremos NNTP en este libro, pero se basa ligeramente en SMTP y tiene un estilo similar.

El protocolo *gopher* corresponde al sistema Gopher, que se diseñó en la Universidad de Minnesota y fue bautizado por los equipos atléticos de la escuela, los Golden Gophers (también es una expresión en jerga que significa "*go for*", es decir, ve y trae). Gopher es más antiguo que Web por varios años. Se trata de un esquema de recuperación de información, conceptualmente parecido a

la Web misma, pero que sólo maneja texto y no imágenes. Ahora es esencialmente obsoleto y no se utiliza con mucha frecuencia.

Los dos últimos protocolos en realidad no obtienen páginas Web, y no son reconocidos por todos los navegadores, pero de todas maneras son útiles. El protocolo *mailto* permite a los usuarios enviar correo electrónico desde un navegador Web. El procedimiento es hacer clic en el botón OPEN y especificar un URL que consista en *mailto:* seguido de la dirección de correo electrónico del destinatario. La mayoría de los navegadores responderá iniciando un programa de correo electrónico con la dirección y algunos de los campos de encabezado ya llenos.

El protocolo telnet sirve para establecer una conexión en línea con una máquina remota. Se utiliza de la misma manera que el programa telnet, lo cual no es sorprendente, puesto que la mayoría de los navegadores simplemente llama al programa telnet como aplicación auxiliar.

En pocas palabras, los URLs se han diseñado no sólo para permitir que los usuarios naveguen por Web, sino para que también utilicen FTP, noticias, Gopher, correo electrónico y telnet; de este modo los programas especializados de interfaz de usuario para esos otros servicios son innecesarios pues casi todos los accesos a Internet se integran en un solo programa, el navegador Web. Si no fuera por el hecho de que este esquema fue diseñado por un investigador de física, podría pensarse fácilmente que lo creó el departamento de publicidad de una compañía de software.

A pesar de todas estas agradables propiedades, el uso creciente de Web ha sacado a la luz una debilidad inherente del esquema URL. Un URL apunta a un *host* específico. En el caso de páginas a las que se hace referencia constante, sería deseable tener varias copias muy distantes, para reducir el tráfico de la red. El problema es que los URLs no proporcionan ninguna manera de referirse a una página sin decir de manera simultánea dónde está. No hay manera de decir: "quiero la página xyz y no me importa de dónde la traigas". Para resolver este problema y hacer posible la duplicación de páginas, la IETF está trabajando en un sistema de **URNs** (**Nombres Universales de Recursos).** Un URN puede considerarse como un URL generalizado. Este tema aún es el objetivo de la investigación, aunque en el RFC 2141 se da una sintaxis propuesta.

Sin estado y cookies

Como hemos visto en varias ocasiones, Web básicamente no tiene estado. No existe el concepto de inicio de sesión. El navegador envía una solicitud a un servidor y obtiene un archivo. A continuación el servidor olvida que ha visto alguna vez a ese cliente en particular.

Al inicio, cuando Web sólo se utilizaba para recuperar documentos disponibles públicamente, este modelo era adecuado. Pero a medida que Web comenzó a adquirir otras funciones, surgieron problemas. Por ejemplo, algunos sitios Web requieren que los clientes se registren (y, con probabilidad, que paguen dinero) para poder utilizarlos. Esto da lugar a la pregunta de cómo los servidores pueden distinguir entre las solicitudes de usuarios registrados y las demás. Un segundo ejemplo es el comercio electrónico. Si un usuario navega en una tienda electrónica y coloca artículos en su carrito de compras de vez en cuando, ¿de qué manera el servidor mantiene un registro del contenido del carrito? Un tercer ejemplo son los portales Web personalizados, como Yahoo.

Los usuarios pueden establecer una página de inicio detallada que contenga sólo la información que desean (por ejemplo, sobre sus acciones y sus equipos de deportes favoritos), pero, ¿de qué manera puede el servidor desplegar la página correcta si no sabe quién es el usuario?

A primera vista, uno podría pensar que los servidores podrían seguir la pista de los usuarios al ver sus direcciones IP. Sin embargo, esta idea no funciona. Primero que nada, muchos usuarios trabajan en computadoras compartidas, especialmente en compañías, y la dirección IP simplemente identifica a la computadora, no al usuario. Segundo, y todavía peor, muchos ISPs utilizan NAT, por lo que todos los paquetes salientes de todos los usuarios tienen la misma dirección IP. Desde el punto de vista del servidor, los miles de clientes de los ISPs utilizan la misma dirección IP.

Para resolver este problema, Netscape diseñó una técnica muy criticada llamada **cookies.** El nombre se deriva de la antigua jerga de programación en la que un programa llama a un procedimiento y obtiene algo similar que tal vez tenga que presentar posteriormente para conseguir que se realice algún trabajo. En este sentido, un descriptor de archivos UNIX o un identificador de objetos de Windows puede considerarse como una *cookie*. Las *cookies* se formalizaron posteriormente en el RFC 2109.

Cuando un cliente solicita una página Web, el servidor puede proporcionar información adicional junto con la página solicitada. Esta información puede incluir una *cookie*, que es un pequeño archivo (o cadena, de a lo mucho 4 KB). Los navegadores almacenan *cookies* ofrecidas en un directorio de *cookies* en el disco duro de la máquina del cliente, a menos que el usuario las haya deshabilitado. Las *cookies* son simplemente archivos o cadenas, no programas ejecutables. En principio, una *cookie* puede contener un virus, pero puesto que las *cookies* se tratan como datos, no hay una forma oficial de que los virus se ejecuten realmente y hagan daño. Sin embargo, siempre es posible que algún *hacker* saque provecho de un error de un navegador para causar la activación de dicho virus.

Una *cookie* puede contener hasta cinco campos, como se muestra en la figura 7-25. El *dominio* indica de dónde viene la *cookie*. Se supone que los navegadores verifican que los servidores no mientan acerca de su dominio. Cada dominio puede almacenar hasta 20 *cookies* por cliente. La *ruta* es la ruta en la estructura del directorio del servidor que identifica qué partes del árbol de archivos del servidor podrían utilizar la *cookie*. Por lo general es /, lo que significa el árbol completo.

Dominio	Ruta	Contenido	Expira	Seguro
toms-casino.com	/	CustomerID=497793521	15-10-02 17:00	Sí
joes-store.com	/	Cart=1-00501;1-07031;2-13721	11-10-02 14:22	No
aportal.com	/	Prefs=Stk:SUNW+ORCL;Spt:Jets	31-12-10 23:59	No
sneaky.com	/	UserID=3627239101	31-12-12 23:59	No

Figura 7-25. Algunos ejemplos de *cookies*.

El campo *Contenido* toma la forma *nombre* = *valor*. Tanto *nombre* como *valor* pueden ser lo que el servidor desee. Este campo es donde se almacena el contenido de la *cookie*.

El campo *Expira* especifica cuándo caduca la *cookie*. Si este campo está ausente, el navegador descarta la *cookie* cuando sale. Tal *cookie* se conoce como ***cookie* no persistente.** Si se proporciona

una hora y una fecha, se dice que la *cookie* es **persistente** y se mantiene hasta que expira. Las fechas de expiración se dan en la hora del Meridiano de Greenwich. Para eliminar una *cookie* del disco duro de un cliente, un servidor simplemente la envía nuevamente, pero con una fecha caducada.

Por último, el campo *Seguro* puede establecerse para indicar que el navegador podría simplemente regresar la *cookie* a un servidor seguro. Esta característica se utiliza para comercio electrónico, actividades bancarias y otras aplicaciones seguras.

Ya vimos cómo se adquieren las *cookies*, pero, ¿cómo se utilizan? Justo antes de que un navegador solicite una página a un sitio Web, verifica su directorio de *cookies* para ver si el dominio al que está solicitando la página ya colocó alguna *cookie*. De ser así, todas las *cookies* colocadas por ese dominio se incluyen en el mensaje de solicitud. Cuando el servidor las obtiene, puede interpretarlas de la forma que desee.

Examinemos algunos usos posibles para las *cookies*. En la figura 7-25, la primera *cookie* fue establecida por *toms-casino.com* y se utiliza para identificar al cliente. Cuando éste inicia una sesión la siguiente semana para despilfarrar más dinero, el navegador envía la *cookie* de forma que el servidor sepa quién es. Una vez que el servidor cuenta con el ID del cliente, puede buscar el registro de éste en una base de datos y utilizar esta información para construir una página Web apropiada para desplegar. Dependiendo de los hábitos conocidos de apuestas del cliente, esta página podría consistir en una mano de póquer, un listado de las carreras de caballos del día o una máquina tragamonedas.

La segunda *cookie* proviene de *joes-store.com*. El escenario aquí es un cliente que está vagando por una tienda en busca de cosas buenas que comprar. Cuando dicho cliente encuentra una ganga y hace clic en ella, el servidor crea una *cookie* que contiene el número de elementos y el código del producto y la regresa al cliente. Conforme el cliente vaga por la tienda, la *cookie* se regresa cada vez que se solicita una página. Conforme se acumulan más compras, el servidor las agrega a la *cookie*. En la figura, el carrito contiene tres elementos, el último de los cuales se requiere dos veces. Por último, cuando el cliente hace clic en PASAR A LA CAJA, la *cookie*, que ahora contiene la lista completa de compras, se envía junto con la solicitud. De esta forma el servidor sabe exactamente qué ha comprado.

La tercera *cookie* es para un portal Web. Cuando el cliente hace clic en un vínculo que lo lleva al portal, el navegador envía la *cookie*. Ésta le indica al portal que construya una página que contenga los precios de las acciones de Sun Microsystems y Oracle, así como los resultados de fútbol de los Jets de Nueva York. Puesto que una *cookie* puede tener un tamaño de hasta 4 KB, hay demasiado espacio para incluir preferencias más detalladas, como encabezados de los periódicos, el clima local, ofertas especiales, etcétera.

Las *cookies* también pueden utilizarse para el beneficio del servidor. Por ejemplo, suponga que un servidor desea llevar el registro de cuántos visitantes únicos ha tenido y cuántas páginas miró cada uno antes de dejar el sitio. Cuando llega la primera solicitud, no hay *cookie* acompañante, por lo que el servidor regresa una *cookie* que contiene *Counter = 1*. Los clics subsecuentes en ese sitio regresarán la *cookie* al servidor. Cada vez el contador se incrementa y se regresa al cliente. Al llevar un registro de los contadores, el servidor puede ver cuántas personas salieron del sitio después de ver la primera página, cuántos vieron dos páginas, y así sucesivamente.

Las *cookies* también han sido objeto de malos usos. En teoría, se supone que las *cookies* sólo deben regresar al sitio original, pero los *piratas informáticos* han aprovechado varios errores de los navegadores para capturar las *cookies* que no son para ellos. Desde que algunos de los sitios de comercio electrónico colocaron números de tarjetas de crédito en las *cookies*, los intentos de abuso han sido más evidentes.

Un uso controversial de las *cookies* es colectar de manera secreta información sobre los hábitos de navegación en Web de los usuarios. Funciona como se explica a continuación. Una agencia de publicidad, digamos, Sneaky Ads, contacta sitios Web grandes y paga a los dueños una cuota por colocar anuncios de los productos de sus clientes corporativos. En lugar de dar al sitio un archivo GIF o JPEG para que lo coloque en cada página, le proporciona un URL para dicho propósito. Cada URL que dicha agencia maneja contiene un número único en la parte del archivo, como

```
http://www.sneaky.com/382674902342.gif
```

Cuando un usuario visita por primera vez una página, *P*, que contiene un anuncio de éstos, el navegador obtiene el archivo HTML. A continuación el navegador inspecciona el archivo HTML y ve el vínculo al archivo de imagen en *www.sneaky.com*, por lo que envía un mensaje en el que solicita la imagen. Se regresa un archivo GIF que contiene un anuncio, junto con una *cookie* que contiene un ID de usuario único, 3627239101 en la figura 7-25. Sneaky registra el hecho de que el usuario con este ID visitó la página *P*. Esto es fácil de hacer puesto que se hace referencia al archivo solicitado (*382674902342.gif*) sólo en la página *P*. Por supuesto, el anuncio real puede aparecer en miles de páginas, pero cada vez con un nombre de archivo diferente. Sneaky probablemente cobre un par de centavos al fabricante del producto cada vez que envíe el anuncio.

Más tarde, cuando el usuario visite otra página Web que contenga cualquiera de los anuncios de Sneaky, después de que el navegador ha obtenido el archivo HTML del servidor, ve el vínculo a, digamos, *http://www.sneaky.com/493654919923.gif* y solicita ese archivo. Puesto que ya tiene una *cookie* del dominio *sneaky.com*, el navegador incluye una *cookie* de Sneaky que contiene el ID del usuario. Sneaky ahora conoce una segunda página que el usuario ha visitado.

A su debido tiempo, Sneaky puede construir un perfil completo de los hábitos de navegación del usuario, aunque éste nunca ha hecho clic en ninguno de los anuncios. Por supuesto, aún no tiene el nombre del usuario (aunque tiene su dirección IP, lo cual debe ser suficiente para deducir el nombre a partir de otras bases de datos). Sin embargo, si el usuario alguna vez proporciona su nombre a algún sitio que coopere con Sneaky, estará disponible un perfil completo junto con un nombre para venderlo a quien desee comprarlo. La venta de esta información puede ser lo suficientemente rentable para que Sneaky coloque más anuncios en más sitios Web y, por lo tanto, para que colecte más información. La parte más insidiosa de este negocio es que la mayoría de los usuarios desconocen por completo esta recolección de información y tal vez piensen que están a salvo porque no hacen clic en ninguno de los anuncios.

Y si Sneaky desea ser supersneaky, el anuncio no necesita ser el clásico. Un “anuncio” que conste de un color de fondo de un solo píxel (por lo que es invisible), funciona exactamente de la misma forma: requiere que el navegador obtenga la imagen gif de 1×1 píxeles y le envíe todas las *cookies* que se originan en el dominio del píxel.

Para mantener algo de su privacidad, algunos usuarios configuran sus navegadores para que rechacen las *cookies*. Sin embargo, esto puede darles problemas con los sitios Web legítimos que utilizan *cookies*. Para resolver este problema, algunas veces los usuarios instalan software que elimina *cookies*. Éstos son programas especiales que inspeccionan cada *cookie* entrante al momento del arribo y la aceptan o descartan, dependiendo de las opciones que el usuario haya establecido (por ejemplo, sobre qué sitios Web pueden ser confiables). Esto le da al usuario un control detallado sobre cuáles *cookies* se aceptan y cuáles se rechazan. Los navegadores modernos, como Mozilla (*www.mozilla.org*), han elaborado controles de usuario sobre *cookies* integradas.

7.3.2 Documentos Web estáticos

La base de Web es la transferencia de páginas Web desde el servidor al cliente. En la forma más simple, las páginas Web son estáticas, es decir, son simplemente archivos que se encuentran en algún servidor esperando a ser recuperados. En este sentido, incluso un vídeo es una página Web estática porque es sólo un archivo. En esta sección examinaremos en detalle las páginas Web estáticas. En la siguiente examinaremos el contenido dinámico.

HTML—Lenguaje de Marcado de Hipertexto

En la actualidad las páginas Web se escriben en un lenguaje llamado **HTML** (**Lenguaje de Marcado de Hipertexto**). HTML permite a los usuarios producir páginas Web que incluyen texto, gráficos y apuntadores a otras páginas Web. HTML es un lenguaje de marcado que sirve para describir cómo se van a formatear los documentos. El término "marcado" proviene de la época en que los correctores de estilo realmente marcaban los documentos para indicar a la imprenta —en aquellos tiempos, un humano— qué fuentes utilizar, y cosas por el estilo. Por lo tanto, los lenguajes de marcado contenían comandos explícitos para formatear. Por ejemplo, en HTML, significa **iniciar modo en negritas y** significa abandonar modo en negritas. La ventaja de un lenguaje de marcado sobre uno con marcado no explícito es que escribir un navegador para él es directo: el navegador simplemente tiene que entender los comandos de marcado. TeX y troff son ejemplos de otros lenguajes de marcado bien conocidos.

Al integrar todos los comandos de marcado dentro de cada archivo HTML y al estandarizarlos, se hace posible que cualquier navegador Web lea y reformatee cualquier página Web. La capacidad de reformatear las páginas Web tras su recepción es crucial porque una página pudo haberse producido en una ventana de 1600×1200 con colores de 24 bits pero tal vez se vaya a desplegar en una de 640×320 con colores de 8 bits.

A continuación daremos una breve introducción al HTML, sólo para dar una idea de lo que es. Aunque ciertamente es posible escribir documentos HTML con cualquier editor estándar (y mucha gente lo hace), también es posible usar editores HTML especiales o procesadores de texto que pueden hacer la mayoría del trabajo (pero que por lo mismo dan al usuario menos control sobre todos los detalles del resultado final).

Una página Web consiste en un encabezado y un cuerpo encerrado entre **etiquetas** (comandos de formateo) <html> y </html>, aunque la mayoría de los navegadores no se quejan si faltan estas etiquetas. Como puede verse en la figura 7-26(a), el encabezado está delimitado por las etiquetas <head> y </head>, y el cuerpo, por las etiquetas <body> y </body>. Los comandos dentro de las etiquetas se llaman **directivas.** La mayoría de las etiquetas HTML tiene este formato, es decir, <algo> para marcar el comienzo de algo, y </algo> para marcar su fin. La mayoría de los navegadores tienen un elemento de menú VIEW SOURCE o algo parecido. Su selección presenta el código fuente HTML de las páginas actuales, en lugar de su salida formateada.

Las etiquetas pueden escribirse tanto en minúsculas como en mayúsculas. Por lo tanto, <head> y <HEAD> significan la misma cosa, pero las nuevas versiones del estándar requieren el uso de minúsculas. La distribución real del documento HTML no es importante. Los analizadores ignoran los espacios extra y los retornos de carro, puesto que tienen que formatear el texto para acomodarlo en el área de presentación. En consecuencia, pueden agregar espacios virtuales a voluntad para hacer más legibles los documentos HTML, algo que la mayoría necesita con urgencia. Como consecuencia, no pueden usarse líneas en blanco para separar párrafos, puesto que simplemente se ignoran. Se requiere una etiqueta específica.

Algunas etiquetas tienen parámetros (nombrados), llamados **atributos.** Por ejemplo,

<img src="abc" alt="foobar">

es una etiqueta, <img>, que tiene el parámetro *SRC* establecido a *abc* y el parámetro *alt* a *foobar*. Para cada etiqueta, el estándar HTML da una lista de los parámetros permitidos, si los hay, y lo que significan. Puesto que cada parámetro se nombra, el orden en que se dan los parámetros no es de importancia.

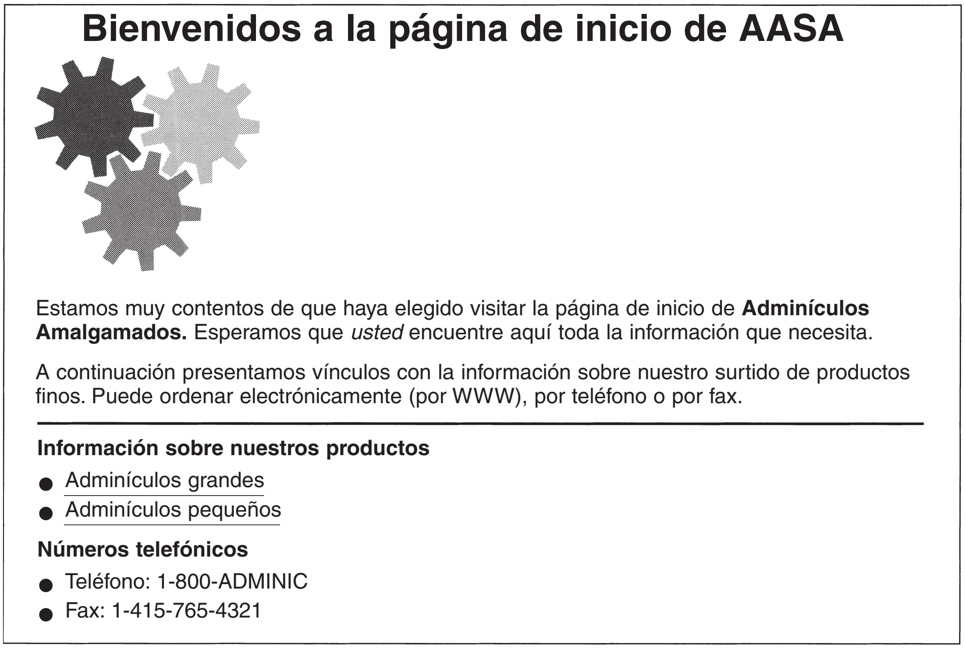
Técnicamente, los documentos HTML se escriben en el conjunto de caracteres Latin-1 del ISO 8859-1, pero para los usuarios cuyos teclados sólo reconocen ASCII, hay secuencias de escape para los caracteres especiales, como è. La lista de caracteres especiales se proporciona en el estándar. Todos estos caracteres comienzan con un signo & y terminan con un punto y coma. Por ejemplo,   produce un espacio, è produce è y é produce é. Puesto que <, >, y & tienen significados especiales, pueden expresarse sólo con sus secuencias de escape, < > y &, respectivamente.

El elemento principal del encabezado es el título, delimitado por <title> y </title>, pero pueden estar presentes también ciertos tipos de metainformación. El título mismo no se representa en la página. Algunos navegadores lo usan para rotular la ventana de la página.

Veamos ahora algunas de las otras características mostradas en la figura 7-26. Todas las etiquetas usadas en esa figura y algunas más se presentan en la figura 7-27. Los encabezados se generan con una etiqueta <h*n*>, donde *n* es un dígito del intervalo 1 a 6. Por lo tanto, <h1> es el encabezado más importante y <h6> es el menos importante. Es responsabilidad del navegador presentarlas de manera adecuada en la pantalla. Comúnmente, los encabezados de número menor se presentarán con una fuente más grande y gruesa. El navegador también puede seleccionar colores distintos para cada nivel de encabezado. Por lo común, los encabezados <h1> son grandes y en negritas, con cuando menos una línea en blanco arriba y abajo. En contraste, los encabezados <h2> tienen una fuente más pequeña y con menos espacio arriba y abajo.

```
<html>
<head> <title> ADMINÍCULOS AMALGAMADOS, S.A. </title> </head>
<body> <h1> Bienvenidos a la página de inicio de AASA </h1>
<img src="http://www.widget.com/images/logo.gif" ALT="AWI Logo"> <br>
Estamos muy contentos de que haya elegido visitar la página de inicio de <b>
Adminículos Amalgamados </b>. Esperamos que <i> usted </i> encuentre aquí toda
la información que necesita.
<p> A continuación presentamos vínculos con la información sobre nuestro
surtido de productos finos. Puede ordenar electrónicamente (por WWW), por
teléfono o por fax. </p>
<hr>
<h2> Información sobre nuestros productos </h2>
<ul>
  <li> <a href="http://widget.com/products/big"> Adminículos grandes </a>
  <li> <a href="http://widget.com/products/little"> Adminículos pequeños </a>
</ul>
<h2> Números telefónicos </h2>
<ul>
  <li> Teléfono: 1-800-ADMINIC
  <li> Fax: 1-415-765-4321
</ul>
</body>
</html>
```

(a)



Bienvenidos a la página de inicio de AASA

Estamos muy contentos de que haya elegido visitar la página de inicio de **Adminículos Amalgamados.** Esperamos que *usted* encuentre aquí toda la información que necesita.

A continuación presentamos vínculos con la información sobre nuestro surtido de productos finos. Puede ordenar electrónicamente (por WWW), por teléfono o por fax.

Información sobre nuestros productos

- Adminículos grandes
- Adminículos pequeños

Números telefónicos

- Teléfono: 1-800-ADMINIC
- Fax: 1-415-765-4321

(b)

Figura 7-26. (a) HTML para una página Web de ejemplo. (b) Página con formato.

Las etiquetas <b> e <i> se usan para entrar en el modo de negritas y el de cursivas, respectivamente. Si el navegador no es capaz de presentar negritas y cursivas debe usar algún otro método para mostrarlas (por ejemplo, un color diferente para cada una o tal vez vídeo inverso).

HTML proporciona varios mecanismos para hacer las listas, entre ellas las listas anidadas. Las listas inician con <ul> u <ol>, y <li> marca el inicio de los elementos en ambos casos. La etiqueta <ul> comienza una lista desordenada. Los elementos individuales, que se marcan con la etiqueta <li> en el origen, aparecen con viñetas (•) al principio. Una variante de ese mecanismo es <ol>, que es para listas ordenadas. Cuando se usa esta etiqueta, los elementos <li> son numerados por el navegador. Aparte del uso de diferentes etiquetas de inicio y fin, <ul> y <ol> tienen la misma sintaxis y resultados parecidos.

Las etiquetas
, <p> y <hr> indican límites entre secciones del texto. El formato preciso se puede determinar por la hoja de estilo (vea más adelante) asociada a la página. La etiqueta
 simplemente fuerza una línea nueva. Por lo común, los navegadores no insertan una línea en blanco tras una
. En contraste, <p> comienza un párrafo y podría, por ejemplo, insertar una línea en blanco y posiblemente una sangría. (En teoría, </p> existe para marcar el fin del párrafo, pero pocas veces se usa; la mayoría de los autores de HTML ni siquiera sabe que existe.) Por último, <hr> fuerza una nueva línea y dibuja una línea horizontal a lo ancho de la pantalla.

HTML permite la inclusión de imágenes en línea en una página Web. La etiqueta <img> especifica que se cargará una imagen en la posición actual de la página; puede tener varios parámetros. El parámetro *src* indica el URL de la imagen. El estándar HTML no especifica los formatos gráficos permitidos. En la práctica, todos los navegadores soportan archivos GIF y JPEG. Los navegadores pueden soportar otros formatos, pero esta extensión es un arma de doble filo. Si un usuario está acostumbrado a un navegador que reconoce, digamos, archivos BMP, puede incluir éstos en sus páginas Web y luego sorprenderse de que otros navegadores simplemente ignoren todo su maravilloso arte.

Otros parámetros de <img> son *align*, que controla la alineación de la imagen con respecto a la línea base del texto (*top, middle, bottom*), *alt*, que proporciona texto a usar en lugar de la imagen cuando el usuario ha inhabilitado las imágenes, e *ismap*, un indicador de que la imagen es un mapa activo (es decir, un elemento en el que se puede hacer clic).

Por último, llegamos a los hipervínculos, que utilizan las etiquetas <a> (ancla) y . Al igual que <img>, <a> tiene varios parámetros, entre los que se encuentran *href* (el URL) y *name* (el nombre del hipervínculo). El texto entre <a> y se despliega. Si se selecciona, se sigue el hipervínculo a una nueva página. También se permite poner una imagen <img> ahí, en cuyo caso hacer clic en la imagen también activa el hipervínculo.

Como ejemplo, considere el siguiente fragmento HTML:

```
<a href="http://www.nasa.gov"> Página de inicio de la NASA </a>
```

Cuando se presenta una página con este fragmento, lo que aparece en la ventana es

<u>Página de inicio de la NASA</u>

Etiqueta	Descripción
<html>...</html>	Declara que la página Web está descrita en HTML
<head>...</head>	Delimita el encabezado de la página
<title>...</title>	Delimita el título (no se presente en la página)
<body>...</body>	Delimita el cuerpo de la página
<h*n*>...</h*n*>	Delimita un encabezado de nivel *n*
...	Pone...en negritas
<i>...</i>	Pone...en cursivas
<center>...</center>	Centra...en la página horizontalmente
...	Corchetes de una lista desordenada (con viñetas)
...	Corchetes de una lista numerada
...	Corchetes de un elemento de una lista ordenada o numerada
 	Obliga salto de línea aquí
<p>	Inicia un párrafo
<hr>	Inserta una regla horizontal
<img src="...">	Carga una imagen aquí
...	Define un hipervínculo

Figura 7-27. Selección de etiquetas HTML comunes. Algunas tienen parámetros adicionales.

Si el usuario hace clic en este texto, el navegador de inmediato trae la página cuyo URL es *http://www.nasa.gov* y la presenta.

Como segundo ejemplo, considere

```
<a href="http://www.nasa.gov"> <img src="shuttle.gif" alt="NASA"> </a>
```

Al presentarse, esta página muestra una fotografía (por ejemplo, del transbordador espacial). Al hacer clic en la foto se pasa a la página de inicio de la NASA, igual que al hacer clic en el texto subrayado del ejemplo anterior. Si el usuario inhabilita la presentación automática de imágenes, el texto NASA se presentará donde iría la fotografía.

La etiqueta <a> puede aceptar un parámetro *name* para plantar un hipervínculo, de modo que pueda hacerse referencia a él desde la página. Por ejemplo, algunas páginas Web comienzan con un índice de materias en el que se puede hacer clic. Al hacer clic en un elemento de dicho índice, el usuario salta a la sección correspondiente de la página.

HTML sigue evolucionando. HTML 1.0 y HTML 2.0 no tenían tablas, pero éstas se agregaron en HTML 3.0. Una tabla HTML consiste en una o más filas, cada una de las cuales contiene una o más **celdas.** Éstas pueden contener una gama amplia de material, como texto, figuras, iconos, fotografías e incluso otras tablas. Es posible combinar las celdas a fin de que, por ejemplo, un encabezado pueda abarcar varias columnas. Los autores de páginas tienen control limitado sobre el diseño —incluyendo la alineación, los estilos de los bordes y los márgenes de las celdas—, pero los navegadores tienen la palabra final al representar las tablas.

En la figura 7-28(a) se lista una definición de tabla HTML y en la figura 7-28(b) se muestra una posible representación. Este ejemplo simplemente muestra algunas de las características básicas de las tablas HTML. Las tablas se inician mediante la etiqueta <table>. Es posible proporcionar información adicional para describir las propiedades generales de la tabla.

La etiqueta <caption> puede utilizarse para proporcionar el título de una figura. Cada fila comienza con una etiqueta <tr> (fila de tabla). Las celdas individuales se marcan como <th> (encabezado de tabla) o <td> (datos de tabla). La distinción se realiza para permitir que los navegadores utilicen diferentes representaciones para ellas, como lo hemos hecho en este ejemplo.

En la tablas también se permiten varios atributos. Incluyen formas de especificar alineaciones de celda horizontales y verticales, texto justificado dentro de una celda, bordes, grupos de celdas, unidades y más.

En HTML 4.0 se agregaron nuevas características. Éstas incluyen características de accesibilidad para usuarios discapacitados, incrustación de objetos (una generalización de la etiqueta <img> de manera que otros objetos también puedan ser incrustados en las páginas), soporte para lenguajes de secuencias de comandos (para permitir contenido dinámico), y más.

Cuando un sitio Web es complejo, y consiste en muchas páginas creadas por diversos autores que trabajan para la misma compañía, con frecuencia es deseable tener una forma de evitar que páginas diferentes tengan apariencias distintas. Este problema puede resolverse utilizando **hojas de estilo.** Con éstas, las páginas individuales ya no utilizan más estilos físicos, como negritas e itálicas. En su lugar, los autores de páginas utilizan estilos lógicos, como <dn> (define), <em> (énfasis débil), <strong> (énfasis fuerte) y <var> (variables de programa). Los estilos lógicos se definen en la hoja de estilo, que se indica al principio de cada página. De esta manera, todas las páginas tienen el mismo estilo, y si el Webmaster decide cambiar <strong> de itálicas de 14 puntos en azul a negritas de 18 puntos en rosa, todo lo que necesita es cambiar una definición para convertir todo el sitio Web. Una hoja de estilo se puede comparar con un archivo #include de un programa C: al cambiar una definición de macro ahí, se cambia esa definición de todos los archivos de programa que incluyen el encabezado.

Formularios

HTML 1.0 básicamente era de un solo sentido. Los usuarios podían llamar las páginas desde los proveedores de información, pero era difícil enviar información en el otro sentido. A medida que más y más organizaciones comerciales comenzaron a utilizar Web, creció la demanda del tráfico de dos vías. Por ejemplo, muchas compañías querían tener la capacidad de tomar pedidos de productos mediante sus páginas Web, los proveedores de software querían distribuir software por medio de Web y hacer que sus clientes llenaran sus tarjetas de registro electrónicamente, y las compañías que ofrecían búsquedas en Web querían que sus clientes tuvieran la capacidad de indicar claves de búsqueda.

Estas demandas condujeron a la inclusión de **formularios** a partir de HTML 2.0. Los formularios contienen cuadros o botones que permiten a los usuarios proporcionar información o tomar decisiones, y después enviar dicha información al dueño de la página. Los formularios utilizan la etiqueta <input> para este propósito. Esta etiqueta tiene una variedad de parámetros para determinar

```
<html>
<head> <title> Una página de ejemplo con una tabla </title> </head>
<body>
<table border=1 rules=all>
<caption> Algunas diferencias entre las versiones de HTML </caption>
<col align=left>
<col align=center>
<col align=center>
<col align=center>
<col align=center>
<tr> <th>Elemento <th>HTML 1.0 <th>HTML 2.0 <th>HTML 3.0 <th>HTML 4.0 </tr>
<tr> <th> Hipervínculos <td> x <td> x <td> x <td> x </tr>
<tr> <th> Imágenes <td> x <td> x <td> x <td> x </tr>
<tr> <th> Listas <td> x <td> x <td> x <td> x </tr>
<tr> <th> Mapas activos e imágenes <td>   <td> x <td> x <td> x </tr>
<tr> <th> Formularios <td>   <td> x <td> x <td> x </tr>
<tr> <th> Ecuaciones <td>   <td>   <td> x <td> x </tr>
<tr> <th> Barras de herramientas <td>   <td>   <td> x <td> x </tr>
<tr> <th> Tablas <td>   <td>   <td> x <td> x </tr>
<tr> <th> Características de accesibilidad <td>   <td>   <td>  
          <td> x </tr>
<tr> <th> Incrustación de objetos <td>   <td>   <td>  
          <td> x </tr>
<tr> <th> Secuencias de comandos <td>   <td>   <td>   <td> x
          </tr>
</table>
</body>
</html>
```

(a)

Algunas diferencias entre las versiones de HTML

Elemento	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0
Hipervínculos	X	X	X	X
Imágenes	X	X	X	X
Listas	X	X	X	X
Mapas activos e imágenes		X	X	X
Formularios		X	X	X
Ecuaciones			X	X
Barras de herramientas			X	X
Tablas			X	X
Características de accesibilidad				X
Incrustación de objetos				X
Secuencias de comandos				X

Figura 7-28. (a) Una tabla HTML. (b) Una posible representación de esta tabla.

el tamaño, la naturaleza y el uso del cuadro presentado. Los formularios más comunes constan de campos en blanco para aceptar texto del usuario, casillas de verificación que pueden marcarse, mapas activos y botones de envío. El ejemplo de la figura 7-29 ilustra algunas de estas opciones.

Comencemos nuestro estudio de los formularios repasando este ejemplo. Como todos los formularios, éste se encierra entre las etiquetas <form> y </form>. El texto no delimitado por etiquetas simplemente se despliega. En un formulario se permiten todas las etiquetas normales (por ejemplo, <b>). Se usan tres tipos de cuadros de entrada en este formulario.

El primer cuadro de entrada está después del texto "Nombre". Dicho cuadro tiene 46 caracteres de longitud y espera que el usuario escriba una cadena, que luego se almacenará en la variable *customer* para procesarse posteriormente. La etiqueta <p> indica al navegador que despliegue el texto y los cuadros subsiguientes en otra línea, aunque haya espacio en la línea actual. Usando <p> y otras etiquetas de distribución, el autor de la página puede controlar la manera en que se ve el formulario en la pantalla.

La siguiente línea del formulario —la cual tiene 40 caracteres de longitud— solicita la dirección del usuario, también en una línea individual. Luego se encuentra una línea que solicita la ciudad, el estado y el país. Aquí no se usan etiquetas <p> entre los campos, por lo que el navegador desplegará todos en la misma línea, si caben. En lo que concierne al navegador, este párrafo simplemente contiene seis elementos: tres cadenas que alternan con tres cuadros. Todo esto lo despliega de manera lineal de izquierda a derecha, pasando a una línea nueva cada vez que la línea actual ya no puede contener el siguiente elemento. Por lo tanto, es concebible que en una pantalla de 1600×1200 las tres cadenas y sus cuadros correspondientes aparecerán en la misma línea, pero en una pantalla de 1024×768 tal vez se dividan en dos líneas. En el peor caso, la palabra "País" está al final de una línea y su cuadro al principio de la siguiente.

La siguiente línea solicita el número de tarjeta de crédito y su fecha de expiración. La transmisión de números de tarjeta de crédito por Internet sólo debe hacerse cuando se han tomado las medidas de seguridad adecuadas. En el capítulo 8 analizaremos algo de esto.

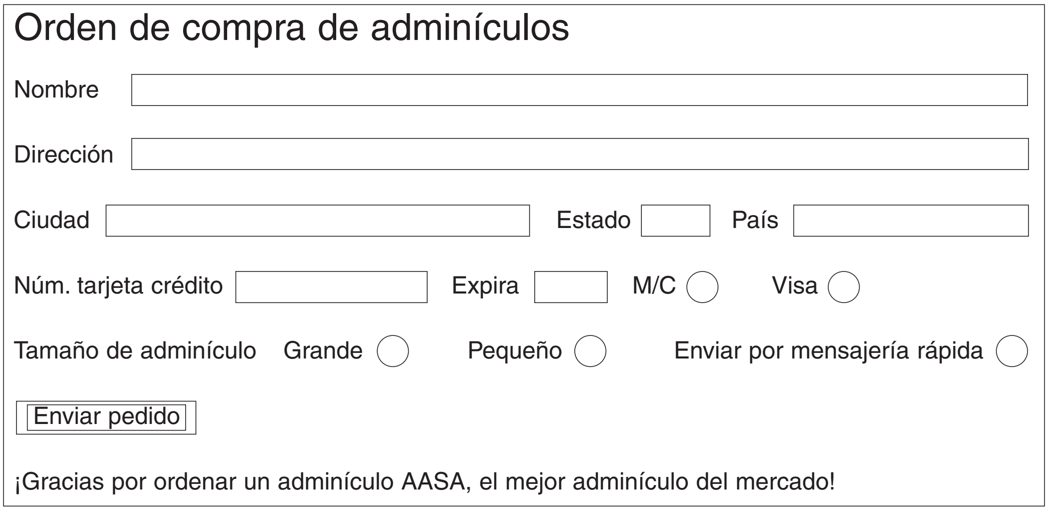
A continuación de la fecha de expiración encontramos una característica nueva: botones de opción. Éstos se usan cuando debe tomarse una decisión entre dos o más alternativas. Aquí el modelo es un radio de automóvil con media docena de botones para elegir estaciones. El navegador presenta estos botones de forma que permite que el usuario los seleccione y deseleccione haciendo clic en ellos (o usando el teclado). Al hacer clic en uno de los botones, se desactivan todos los demás del mismo grupo. La presentación visual depende del navegador. El tamaño del adminículo también usa dos botones de opción. Los dos grupos se distinguen por su campo *name*, no por el alcance estático usando algo como <radiobutton>...</radiobutton>.

Los parámetros de *value* sirven para indicar el botón de opción en el que se ha hecho clic. Dependiendo de las opciones de tarjeta de crédito que haya seleccionado el usuario, a la variable *cc* se le asignará la cadena "mastercard" o la cadena "visacard".

Después de los dos grupos de botones de opción, llegamos a la opción de envío, representada por un cuadro de tipo *checkbox*, que puede estar activo o inactivo. A diferencia de los botones de opción, de los que sólo se puede seleccionar uno del grupo, cada cuadro de tipo *checkbox* puede estar activo o inactivo, independientemente de los demás. Por ejemplo, al ordenar una pizza mediante la página Web de Electropizza, el usuario puede seleccionar sardinas *y* cebolla *y* piña

```
<html>
<head> <title> FORMULARIO DE PEDIDO DE CLIENTES AASA </title> </head>
<body>
<h1> Orden de compra de adminículos </h1>
<form ACTION="http://widget.com/cgi-bin/widgetorder" method=POST>
<p> Nombre <input name="customer" size=46> </p>
<p> Dirección <input name="address" size=40> </p>
<p> Ciudad <input name="city" size=20> Estado <input name="state" size =4>
País <input name="country" size=10> </p>
<p> Núm. tarjeta crédito <input name="cardno" size=10>
Expira <input name="expires" size=4>
mastercard <input name="cc" type=radio value="mastercard">
visacard <input name="cc" type=radio value="visacard"> </p>
<p> Tamaño de adminículo Grande <input name="product" type=radio
    value="expensive">
Pequeño <input name="product" type=radio value="cheap">
Enviar por mensajería rápida <input name="express" type=checkbox> </p>
<p><input type=submit value="Enviar pedido"> </p>
¡Gracias por ordenar un adminículo AASA, el mejor adminículo del mercado!
</form>
</body>
</html>
```

(a)



(b)

Figura 7-29. (a) Código HTML para un formulario de pedido. (b) Página formateada.

(si se atreve), pero no puede seleccionar pequeña *y* mediana *y* grande para la misma pizza. Los ingredientes de la pizza se representarían con tres cuadros independientes del tipo *checkbox*, mientras que el tamaño de la pizza sería un grupo de botones de opción.

Como nota, en las listas muy largas en las que debe hacerse una selección, los botones de opción son poco prácticos. Por lo tanto, se proporcionan las etiquetas <select> y </select> para delimitar una lista de alternativas, pero con la semántica de los botones de opción (a menos que se dé el parámetro *multiple*, en cuyo caso la semántica es la de las casillas de verificación). Algunos navegadores presentan los elementos entre <select> y </select> como menú desplegable.

Hemos visto dos de los tipos integrados de la etiqueta <input>: *radio* y *checkbox*. De hecho, ya hemos visto también un tercero: *text*. Como este tipo es el predeterminado, no nos molestamos en incluir el parámetro *type* = *text*, pero podríamos haberlo hecho. Otros dos tipos son *password* y *textarea*. Un cuadro *password* es igual a uno *text*, excepto que los caracteres no se despliegan cuando se escriben. Un cuadro *textarea* es igual a un cuadro *text*, excepto que puede contener varias líneas.

Regresando al ejemplo de la figura 7-29, ahora nos topamos con un ejemplo de botón de envío. Al hacer clic en él, la información del usuario introducida en el formulario se envía a la máquina que proporcionó dicho formulario. Al igual que los demás tipos, *submit* es una palabra reservada que el navegador entiende. La cadena *value* es la etiqueta del botón, la cual se despliega. Todos los cuadros pueden tener valores; únicamente necesitamos esa característica aquí. En el caso de los cuadros *text*, el contenido del campo *value* se presenta junto con el formulario, pero el usuario puede modificarlo o borrarlo. Los cuadros *checkbox* y *radio* también pueden inicializarse, pero con un campo llamado *checked* (puesto que *value* simplemente da el texto, pero no indica una selección preferida).

Cuando el usuario hace clic en el botón de envío, el navegador empaca la información recolectada en una línea larga y la regresa al servidor para que la procese. El & se utiliza para separar campos y + para representar un espacio. Para nuestro formulario de ejemplo, la línea podría parecerse al contenido de la figura 7-30 (aquí se divide en tres líneas porque la página no es lo suficientemente ancha):

```
customer=John+Doe&address=100+Main+St.&city=White+Plains&
state=NY&country=USA&cardno=1234567890&expires=6/98&cc=mastercard&
product=cheap&express=on
```

Figura 7-30. Una posible respuesta del navegador al servidor con información introducida por el usuario.

La cadena regresaría al servidor como una línea, no tres. Si no se selecciona un *checkbox*, se omite de la cadena. Es responsabilidad del servidor darle sentido a esta cadena. Más adelante en este capítulo analizaremos cómo puede hacerse esto.

XML y XSL

HTML, con o sin formularios, no proporciona estructura alguna para las páginas Web. Además, mezcla el contenido con el formato. Conforme el comercio electrónico y otras aplicaciones se vuelven más comunes, hay una necesidad cada vez mayor de dar estructura a las páginas Web y de separar el contenido del formato. Por ejemplo, un programa que busca en Web el mejor precio de algún libro o CD necesita analizar muchas páginas Web en busca del título y precio del elemento. Con las páginas Web en HTML, es muy difícil que un programa averigüe en dónde están el título y el precio.

Por esta razón, el W3C ha mejorado el HTML para permitir que las páginas Web tengan estructura para su procesamiento automatizado. Para este propósito se han desarrollado dos nuevos lenguajes. El primero, **XML** (**Lenguaje de Marcado Extensible**), describe el contenido Web de una forma estructurada y el segundo, **XSL** (**Lenguaje de Hojas de estilo Extensible**), describe el formato independientemente del contenido. Estos dos lenguajes son temas extensos y complicados, por lo que nuestra breve introducción sólo rasca la superficie, pero debe darle una idea de cómo funcionan.

Considere el documento XML de ejemplo que se muestra en la figura 7-31. Define una estructura llamada book_list, que es una lista de libros. Cada libro tiene tres campos: el título, autor y año de publicación. Estas estructuras son muy sencillas. Está permitido tener estructuras con campos repetidos (por ejemplo, varios autores), opcionales (como el título del CD-ROM incluido) y alternativos (por ejemplo, el URL de una librería si el libro está en venta o uno de un sitio de subastas si se encuentra ahí debido a que es el último ejemplar).

En este ejemplo, cada uno de los tres campos es una entidad indivisible, pero sí está permitido dividirlos aún más. Por ejemplo, el campo de autor pudo haberse hecho como se muestra a continuación para proporcionar un control detallado sobre la búsqueda y el formato:

```
<author>
   <first_name> Andrew </first_name>
   <last_name> Tanenbaum </last_name>
</author>
```

Cada campo puede dividirse en subcampos y en subsubcampos, y así sucesivamente.

Todo lo que hace el archivo de la figura 7-31 es definir una lista de libros que contiene tres libros. No dice nada sobre cómo desplegar la página Web en la pantalla. Para proporcionar la información de formato, necesitamos un segundo archivo, *book_list.xsl*, que contiene la definición XSL. Este archivo es una hoja de estilo que indica cómo desplegar la página. (Hay alternativas a las páginas de estilo, como una forma de convertir XML en HTML, pero están más allá del alcance de este libro.)

En la figura 7-32 se muestra un archivo XSL para formatear el de la figura 7-31. Después de algunas declaraciones necesarias, entre ellas el URL del estándar XSL, el archivo contiene etiquetas que comienzan con <html> y <body>. Éstas definen el inicio de la página Web, de la forma

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="book_list.xsl"?>

<book_list>

<book>
  <title> Redes de computadoras, 4/e </title>
  <author> Andrew S. Tanenbaum </author>
  <year> 2003 </year>
</book>

<book>
  <title> Sistemas operativos modernos, 2/e </title>
  <author> Andrew S. Tanenbaum </author>
  <year> 2001 </year>
</book>

<book>
  <title> Organización estructurada de computadoras, 4/e </title>
  <author> Andrew S. Tanenbaum </author>
  <year> 1999 </year>
</book>

</book_list>
```

Figura 7-31. Una página Web simple en XML.

común. Después está la definición de una tabla, que incluye los encabezados para las tres columnas. Observe que además de las etiquetas <th> también hay etiquetas </th>, algo con lo que no nos molestaremos por el momento. Las especificaciones XML y XSL son mucho más estrictas que la especificación HTML. Dichas especificaciones declaran que es obligatorio rechazar archivos incorrectos sintácticamente, aunque el navegador pueda determinar lo que quiso decir el diseñador Web. Un navegador que acepta un archivo XML o XSL incorrecto sintácticamente y repara los errores él mismo no se apega a las especificaciones y se rechazará en una prueba de conformidad. Sin embargo, los navegadores pueden determinar con precisión el error. Esta medida algo draconiana es necesaria para tratar con el inmenso número de páginas Web mal diseñadas que existen actualmente.

La instrucción

```
<xsl:for-each select="book_list/book">
```

es análoga a una instrucción for de C. Causa que el navegador itere por el cuerpo del ciclo (el cual termina con <xsl:for-each>), una iteración por cada libro. Cada iteración envía cinco líneas: <tr>, el título, autor y año, y </tr>. Después del ciclo, se envían las etiquetas de cierre </body> y </html>. El

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     version="1.0">
<xsl:template match="/">

<html>
<body>

<table border="2">
 <tr>
  <th> Título</th>
  <th> Autor</th>
  <th> Año </th>
 </tr>

 <xsl:for-each select="book_list/book">
 <tr>
  <td> <xsl:value-of select="title"/> </td>
  <td> <xsl:value-of select="author"/> </td>
  <td> <xsl:value-of select="year"/> </td>
 </tr>
 </xsl:for-each>
</table>

</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Figura 7-32. Una hoja de estilo XSL.

resultado de que el navegador interprete esta hoja de estilo es el mismo que si la página Web contuviera la tabla en línea. Sin embargo, en este formato los programas pueden analizar el archivo XML y encontrar fácilmente libros publicados después del 2000, por ejemplo. Vale la pena enfatizar que aunque nuestro archivo XSL contenía una clase de ciclo, las páginas Web en XML y XSL aún son estáticas, al igual que las páginas HTML, debido a que simplemente contienen instrucciones para que el navegador despliegue la página, tal como lo hacen las páginas HTML. Por supuesto, para utilizar XML y XSL, el navegador debe tener la capacidad de interpretarlos, aunque la mayoría de los navegadores sí la tienen. Aún no es claro si XSL quitará el control a las hojas de estilo tradicionales.

No hemos mostrado cómo hacer esto, pero XML permite que el diseñador del sitio Web realice archivos de definición en los que las estructuras se definen por adelantado. Estos archivos pueden incluirse, por lo que pueden utilizarse para construir páginas Web complejas. Para obtener información adicional sobre ésta y otras características de XML y XSL, vea alguno de los muchos libros escritos sobre ellos. Dos ejemplos son (Livingston, 2002, y Williamson, 2001).

Antes de terminar el análisis de XML y XSL, vale la pena hacer algunos comentarios sobre la batalla ideológica entre el consorcio de WWW y la comunidad de diseñadores Web. El

objetivo original de HTML era especificar la *estructura* del documento, no su *apariencia*. Por ejemplo,

```
<h1> Fotos de Deborah </h1>
```

indica al navegador que resalte el encabezado, pero no dice nada sobre el tipo, color ni tamaño de fuente. Eso se dejó al navegador, el cual conocía las propiedades del despliegue (por ejemplo, con cuántos píxeles contaba). Sin embargo, muchos diseñadores de páginas Web deseaban control absoluto sobre la forma en que aparecían las páginas, por lo que se agregaron nuevas etiquetas al HTML para controlar la apariencia, como

```
<font face="helvetica" size="24" color="red"> Fotos de Deborah </font>
```

Además se agregaron formas para controlar la posición precisa en la pantalla. El problema con este enfoque es que no es portable. Aunque una página puede desplegarse perfectamente en el navegador en el que se desarrolló, en otro navegador u otra versión del mismo navegador o a una resolución de pantalla diferente, puede ser un problema. XML era en parte un intento por regresar al objetivo original de especificar sólo la estructura, no la apariencia de un documento. Sin embargo, XSL también se proporciona para manejar la apariencia. Sin embargo, ambos formatos pueden utilizarse de manera inadecuada. Puede apostarlo.

Además de describir páginas Web, XML también sirve para otros propósitos. Por ejemplo, se puede utilizar como lenguaje para la comunicación entre programas de aplicación. En particular, **SOAP** (**Protocolo Simple de Acceso a Objetos**) es una forma de realizar RPCs entre aplicaciones en forma independiente del lenguaje y de la aplicación. El cliente construye la solicitud como un mensaje XML y lo envía al servidor, utilizando el protocolo HTTP (descrito a continuación). El servidor envía una respuesta como un mensaje XML formateado. De esta manera, las aplicaciones de plataformas heterogéneas pueden comunicarse.

XHTML—Lenguaje de Marcado de Hipertexto Extendido

HTML sigue evolucionando para satisfacer las nuevas demandas. Muchas personas de la industria sienten que en el futuro la mayoría de los dispositivos habilitados para Web no serán PCs, sino dispositivos tipo PDA de mano inalámbricos. Estos dispositivos tienen memoria limitada para navegadores grandes llenos de heurística que tratan de alguna manera de lidiar con las páginas Web incorrectas sintácticamente. Por lo tanto, el siguiente paso después de HTML 4 es un lenguaje muy exigente. Se llama **XHTML** (**Lenguaje de Marcado de Hipertexto Extendido**) y no HTML 5 porque es esencialmente HTML 4 reformulado en XML. Con esto queremos decir que las etiquetas como <h1> no tienen significado intrínseco. Para obtener el efecto HTML 4, se necesita una definición en el archivo XSL. XHTML es el nuevo estándar Web y se debe utilizar para todas las páginas Web nuevas a fin de alcanzar la máxima portabilidad entre plataformas y navegadores.

Hay seis diferencias mayores y una variedad de diferencias menores entre XHTML y HTML 4. Veamos primero las principales diferencias. Primero, las páginas XHTML y los navegadores

deben apegarse estrictamente al estándar. No más páginas Web de mala calidad. Esta propiedad se heredó de XML.

Segundo, todas las etiquetas y los atributos deben estar en minúsculas. Las etiquetas como <HTML> no son válidas en XHTML. Ahora el uso de etiquetas como <html> es obligatorio. De manera similar, <img SRC="ima001.jpg"> también está prohibido porque contiene un atributo en mayúsculas.

Tercero, ahora se requiere el uso de etiquetas de cierre, incluso para </p>. Las etiquetas que no tienen etiqueta de cierre natural, como
, <hr> e <img>, deben tener una diagonal antes del paréntesis angular de cierre ">", por ejemplo

```
<img src="ima001.jpg" />
```

Cuarto, los atributos deben estar encerrados entre comillas. Por ejemplo,

```
<img SRC="ima001.jpg" height=500 />
```

ya no se permite. El número 500 debe estar encerrado entre comillas, de la misma forma que el nombre del archivo JPEG, aunque 500 sea un número.

Quinto, las etiquetas deben estar anidadas de manera apropiada. En el pasado, el anidamiento correcto no era requerido siempre y cuando el estado final alcanzado fuera el correcto. Por ejemplo,

```
<center> <b> Fotos de vacaciones </center> </b>
```

era legal. En XHTML no lo es. Las etiquetas deben cerrarse en el orden inverso en el que se abrieron.

Sexto, cada documento debe especificar su tipo de documento. Este criterio se aplicó en la figura 7-32. Para un análisis de todos los cambios, mayores y menores, vea *www.w3.org.*

7.3.3 Documentos Web dinámicos

Hasta ahora el modelo que hemos utilizado es el de la figura 6-6: el cliente envía un nombre de archivo al servidor, el cual regresa el archivo. En los primeros días de Web, todo el contenido era, de hecho, estático como éste (sólo archivos). Sin embargo, en los años recientes, cada vez más contenido es dinámico, es decir, se genera a solicitud, en lugar de almacenarlo en disco. La generación de contenido puede suceder ya sea en el servidor o en el cliente. A continuación examinaremos cada uno de estos casos.

Generación de páginas Web dinámicas en el servidor

Para ver por qué es necesaria la generación de contenido en el servidor, considere el uso de formularios, como se describió anteriormente. Cuando un usuario llena un formulario y hace clic en el botón de envío, se envía un mensaje al servidor con el contenido del formulario, junto con los campos que el usuario llenó. Este mensaje no es el nombre de un archivo a regresar. Lo que se necesita es proporcionar el mensaje a un programa o a una secuencia de comandos para que lo procesen. Por lo general, el procesamiento involucra el uso de la información proporcionada por el usuario para buscar un registro en una base de datos del disco del servidor y generar una página HTML personalizada para regresarla al cliente. Por ejemplo, en una aplicación de comercio

electrónico, después de que el usuario hace clic en *PASAR A LA CAJA*, el navegador regresa la *cookie* con el contenido del carrito de compras, pero es necesario invocar a algún programa o secuencia de comandos en el servidor para procesar la *cookie* y generar una página HTML en respuesta. La página HTML podría desplegar un formulario que contenga la lista de los elementos del carrito y la última dirección de envío conocida del usuario junto con una respuesta para verificar la información y especificar la forma de pago. En la figura 7-33 se muestran los pasos necesarios para procesar la información de un formulario HTML.

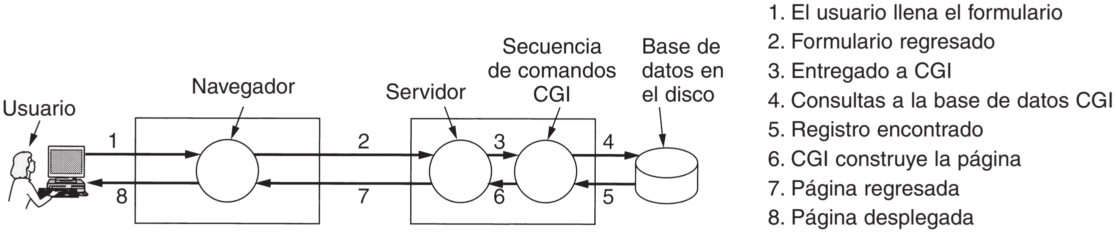


Figura 7-33. Pasos en el procesamiento de información de un formulario HTML.

La forma tradicional de manejar formularios y otras páginas Web interactivas es un sistema llamado **CGI** (**Interfaz de Puerta de Enlace Común).** Es una interfaz estandarizada para permitir que los servidores Web hablen con los programas *back-end* y las secuencias de comandos puedan aceptar datos de entrada (por ejemplo, de formularios) y generar en respuesta páginas HTML. Por lo general, estos *back-ends* son secuencias de comandos escritas en el lenguaje Perl porque las secuencias de comandos de Perl son más fáciles y rápidas de escribir que los programas (si usted sabe programar en Perl). Por convención, se encuentran en un directorio llamado *cgi-bin*, que se muestra en el URL. Algunas veces se utiliza Python en lugar de Perl.

Como ejemplo de la forma en que funciona CGI, considere el caso de un producto de la compañía Productos Verdaderamente Geniales que viene sin una tarjeta de registro de garantía. En su lugar, se le dice al cliente que vaya a *www.pvg.com* para registrarse en línea. En esa página hay un hipervínculo que dice

Haga clic aquí para registrar su producto

Este vínculo apunta a una secuencia de comandos de Perl, digamos, *www.pvg.com/cgi-bin/reg.perl*. Cuando esta secuencia de comandos se invoca sin parámetros, regresa una página HTML que contiene el formulario de registro. Cuando el usuario llena el registro y hace clic en el botón de envío, se regresa un mensaje a esta secuencia de comandos que contiene los valores introducidos y que utiliza el estilo de la figura 7-30. A continuación la secuencia de comandos de Perl analiza los parámetros, crea una entrada en la base de datos para el nuevo cliente y regresa una página HTML que proporciona un número de registro y un número telefónico para el escritorio de ayuda. Ésta no es la única forma de manejar formularios, pero sí es común. Hay muchos libros acerca de cómo crear secuencias CGI y cómo programar en Perl. Algunos ejemplos son (Hanegan, 2001; Lash, 2002, y Meltzer y Michalski, 2001).

Las secuencias de comandos CGI no son la única forma de generar contenido dinámico en el servidor. Otra forma común es incrustar pequeñas secuencias de comandos dentro de páginas HTML y hacer que el servidor mismo sea quien las ejecute para generar la página. Un lenguaje popular para escribir estas secuencias de comandos es **PHP** (**Preprocesador de Hipertexto**). Para utilizarlo, el servidor tiene que entender PHP (de la misma forma que un navegador tiene que entender XML para interpretar páginas Web escritas en XML). Por lo general, los servidores esperan páginas Web que contienen PHP para tener una extensión *php* en lugar de *html* o *htm*.

En la figura 7-34 se ilustra una pequeña secuencia de comandos PHP; debe funcionar con cualquier servidor que tenga instalado PHP. Contiene HTML normal, excepto por la secuencia de comandos PHP dentro de la etiqueta `<?php ... ?>`. Lo que hace es generar una página Web que indica lo que sabe sobre el navegador que la invoca. Normalmente, los navegadores envían alguna información junto con su solicitud (y cualesquier *cookies*) y esta información se coloca en la variable *HTTP_USER_AGENT*. Cuando este listado se coloca en un archivo *test.php* en el directorio WWW de la compañía ABCD, al escribir el URL *www.abcd.com/test.php* se producirá una página Web que indica al usuario cuál navegador, lenguaje y sistema operativo está utilizando.

```
<html>
<body>

<h2> Esto es lo que sé de ti </h2>
<?php echo $HTTP_USER_AGENT ?>

</body>
</html>
```

Figura 7-34. Una página HTML de ejemplo con PHP incrustado.

PHP es especialmente bueno para manejar formularios y es más sencillo que utilizar una secuencia de comandos CGI. Como ejemplo de cómo funciona con formularios, considere la figura 7-35(a). Ésta contiene una página normal HTML con un formulario en ella. Lo único inusual es la primera línea que especifica que se va a invocar el archivo *action.php* para manejar los parámetros después de que el usuario ha llenado y emitido el formulario. La página despliega dos cuadros de texto, uno que solicita un nombre y otro que solicita una edad. Una vez que se han llenado los cuadros de texto y se ha enviado el formulario, el servidor analiza la cadena de tipo regresada de la figura 7-30, colocando el nombre en la variable *nombre* y la edad en la variable *edad*. Después comienza el procesamiento del archivo *action.php*, que se muestra como respuesta en la figura 7-35(b). Durante el procesamiento de este archivo, se ejecutan los comandos PHP. Si el usuario introdujo “Barbara” y “24” en los cuadros, el archivo HTML regresado será el que se muestra en la figura 7-35(c). Por lo tanto, el manejo de formularios con PHP se vuelve muy sencillo.

Aunque PHP es fácil de utilizar, realmente es un lenguaje de programación poderoso diseñado para interactuar entre Web y una base de datos del servidor. Tiene variables, cadenas, arreglos y la mayoría de las estructuras de control que se encuentran en C, pero la E/S es mucho más poderosa que *printf*. PHP es código *open source* y está disponible de manera gratuita.

```
<html>
<body>
<form action="action.php" method="post">
<p> Por favor introduzca su nombre: <input type="text" name="name"> </p>
<p> Por favor introduzca su edad: <input type="text" name="age"> </p>
<input type="submit">
</form>
</body>
</html>
```

(a)

```
<html>
<body>
<h1> Respuesta </h1>
Hola <?php echo $name; ?>.
Predicción: el siguiente año tendrá <?php echo $age + 1; ?>
</body>
</html>
```

(b)

```
<html>
<body>
<h1> Respuesta: </h1>
Hola Barbara.
Predicción: el siguiente año tendrá 25
</body>
</html>
```

(c)

Figura 7-35. (a) Una página Web que contiene un formulario. (b) Una secuencia de comandos PHP para manejar la salida del formulario. (c) Salida de la secuencia de comandos PHP cuando las entradas son “Barbara” y 24, respectivamente.

Se diseñó específicamente para trabajar bien con Apache, que también es código abierto y es el servidor Web más ampliamente utilizado del mundo. Para mayor información sobre PHP, vea (Valade, 2002).

Ya vimos dos formas diferentes para generar páginas HTML dinámicas: las secuencias de comandos CGI y el PHP incrustado. También hay una tercera técnica, llamada **JSP** (**Java Server Pages**), que es similar a PHP, excepto que la parte dinámica se escribe en el lenguaje de programación Java en lugar de en PHP. Las páginas que utilizan esta técnica tienen la extensión de archivo *jsp*. Una cuarta técnica, **ASP** (**Active Server Pages**), es la versión de Microsoft de PHP y JavaServer Pages. Para generar el contenido dinámico utiliza el lenguaje de secuencias de comandos propietario de Microsoft, Visual Basic Script. Las páginas que utilizan esta técnica tienen la extensión *asp*. Por lo general, la selección entre *PHP*, *JSP* y *ASP* tiene que ver más con políticas

(código abierto contra Sun contra Microsoft) que con la tecnología, puesto que estos tres lenguajes son más o menos similares.

La colección de tecnologías para generar contenido al vuelo algunas veces se llama **HTML dinámico.**

Generación de páginas Web dinámicas en el cliente

Las secuencias de comandos de CGI, PHP, JSP y ASP resuelven el problema de manejar formularios e interacciones con bases de datos en el servidor. Pueden aceptar información entrante de formularios, buscar información en una o más bases de datos y generar páginas HTML con los resultados. Lo que ninguno de ellos puede hacer es responder a los movimientos del ratón o interactuar de manera directa con los usuarios. Para esto es necesario tener secuencias de comandos incrustadas en páginas HTML que se ejecuten en la máquina cliente y no en el servidor. Comenzando con HTML 4.0, tales secuencias de comandos son posibles mediante la etiqueta <script>. El lenguaje de secuencias de comandos más popular para el cliente es **JavaScript,** por lo que ahora le daremos un vistazo.

JavaScript es un lenguaje de secuencias de comandos, inspirado por algunas ideas del lenguaje de programación Java. Definitivamente éste no es Java. Al igual que otros lenguajes de secuencias de comandos, es un lenguaje de muy alto nivel. Por ejemplo, en una sola línea de JavaScript es posible desplegar un cuadro de diálogo, esperar entrada de texto y almacenar la cadena resultante en una variable. Las características de alto nivel como éstas hacen que JavaScript sea ideal para diseñar páginas Web interactivas. Por otro lado, el hecho de que no está estandarizado y de que tiene más mutaciones que una mosca de fruta atrapada en una máquina de rayos X, hace extremadamente difícil escribir programas de JavaScript que funcionen en todas las plataformas, pero tal vez algún día se estabilice.

Como ejemplo de un programa de JavaScript, considere el de la figura 7-36. Al igual que el de la figura 7-35(a), despliega un formulario que pide un nombre y una edad, y después calcula cuántos años tendrá una persona el siguiente año. El cuerpo es casi el mismo que el del ejemplo de PHP, la diferencia principal es la declaración del botón de envío y su instrucción de asignación. Ésta indica al navegador que invoque la secuencia de comandos *response* mediante un clic del botón y que la pase al formulario como un parámetro.

Lo que es completamente nuevo aquí es la declaración de la función *response* de JavaScript del encabezado del archivo HTML, un área que se reserva normalmente para títulos, colores de fondo, etcétera. Esta función extrae el valor del campo *name* del formulario y lo almacena como una cadena en la variable *person*. También extrae el valor del campo *age*, lo convierte en un entero mediante el uso de la función *eval*, le agrega 1 y almacena el resultado en *years*. A continuación abre un documento para salida, utiliza el método *writeln* para realizar cuatro escrituras en dicho documento y, por último, lo cierra. Este documento es un archivo HTML, como puede advertirse a partir de las etiquetas HTML. A continuación el navegador despliega el documento en la pantalla.

Es muy importante hacer notar que si bien las figuras 7-35 y 7-36 son muy similares, se procesan de manera totalmente diferente. En la figura 7-35, después de que el usuario hace clic en el botón de envío, el navegador colecta la información en una cadena larga como la de la figura 7-30

```
<html>
<head>
<script language="javascript" type="text/javascript">
function response(test_form) {
  var person = test_form.name.value;
  var years = eval(test_form.age.value) + 1;
  document.open();
  document.writeln("<html> <body>");
  document.writeln("Hola" + person + ".<br>");
  document.writeln("Predicción: el siguiente año usted tendrá" +
    años + ".");
  document.writeln("</body> </html>");
  document.close();
}
</script>
</head>

<body>
<form>
Por favor, introduzca su nombre: <input type="text" name="name">
<p>
Por favor, introduzca su edad: <input type="text" name="age">
<p>
input type="button" value="submit" onclick="response(this.form)">
</form>
</body>
</html>
```

Figura 7-36. Uso de JavaScript para procesar un formulario.

y la regresa al servidor que envió la página. Éste ve el nombre del archivo PHP y lo ejecuta. La secuencia de comandos de PHP produce una nueva página HTML y ésta se regresa al navegador para que la despliegue. En la figura 7-36, cuando se hace clic en el botón de envío, el navegador interpreta una función de JavaScript que está contenida en la página. Todo el trabajo se lleva a cabo de manera local, dentro del navegador. No hay contacto con el servidor. Debido a esto, el resultado se despliega casi instantáneamente, mientras que con PHP, puede haber un retraso de algunos segundos antes de que el HTML resultante llegue al cliente. En la figura 7-37 se ilustra la diferencia entre las secuencias de comandos del servidor y las del cliente, además de los pasos involucrados. En ambos casos, los pasos numerados comienzan después de que se despliega el formulario. El paso 1 consiste en aceptar la entrada del usuario. A continuación se realiza el procesamiento de la entrada, lo cual es diferente en los dos casos.

Esta diferencia no significa que JavaScript sea mejor que PHP. Sus usos son completamente diferentes. PHP (y, por ende, JSP y ASP) se utiliza cuando es necesaria la interacción con una base de datos remota. JavaScript se utiliza cuando la interacción es con el usuario en la máquina

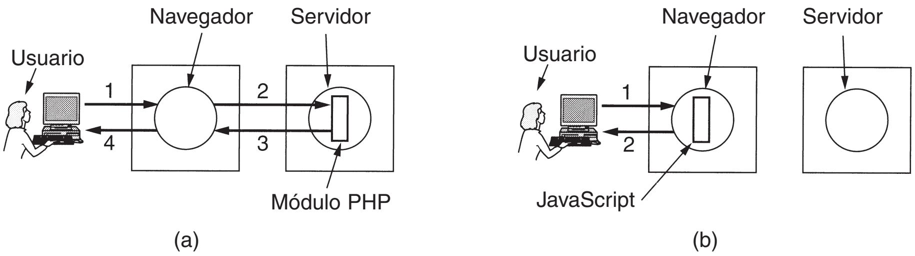


Figura 7-37. (a) Secuencias de comandos en el servidor con PHP. (b) Secuencias de comandos en el cliente con JavaScript.

cliente. Ciertamente es posible (y común) tener páginas HTML que utilicen PHP y JavaScript, aunque éstas no pueden hacer el mismo trabajo ni poseer, por supuesto, el mismo botón.

JavaScript es un lenguaje de programación completo, con todo el poder de C o Java. Tiene variables, cadenas, arreglos, objetos, funciones y todas las estructuras de control comunes. También tiene una gran cantidad de características específicas para páginas Web, entre las que se encuentran la capacidad para administrar ventanas y marcos, establecer y obtener *cookies*, manejar formularios e hipervínculos. En la figura 7-38 se proporciona un programa JavaScript que utiliza una función recursiva.

```
<html>
<head>
<script language="javascript" type="text/javascript">

function response(test_form) {
  function factorial(n) {if (n == 0) return 1; else return n * factorial
    (n - 1);}
  var r = eval(test_form.number.value);    // r = escribe un argumento
  document.myform.mytext.value = "Aquí están los resultados.\n";
  for (var i = 1; i <= r; i++)             // imprime una línea de 1 a r
    document.myform.mytext.value += (i + "! = " + factorial(i) + "\n");
}
</script>
</head>

<body>
<form name="myform">
Please enter a number: <input type="text" name="number">
<input type="button" value="calcular la tabla de factoriales" onclick="
  response(this.form)">
<p>
<textarea name="mytext" rows=25 cols=50> </textarea>
</form>
</body>
</html>
```

Figura 7-38. Un programa JavaScript para calcular e imprimir factoriales.

JavaScript también puede rastrear el movimiento del ratón sobre objetos de la pantalla. Muchas páginas Web de JavaScript tienen la propiedad de que cuando el cursor del ratón se mueve sobre algún texto o imagen, algo pasa. Con frecuencia la imagen cambia o aparece de repente un menú. Este tipo de comportamiento es fácil de programar en JavaScript y produce páginas Web animadas. En la figura 7-39 se muestra un ejemplo.

```
<html>
<head>
<script language="javascript" type="text/javascript">
if (!document.myurl) document.myurl = new Array();
document.myurl[0] = "http://www.cs.vu.nl/ast/im/gato.jpg";
document.myurl[1] = "http://www.cs.vu.nl/ast/im/perro.jpg";
document.myurl[2] = "http://www.cs.vu.nl/ast/im/conejo.jpg";
function pop(m) {
  var urx = "http://www.cs.vu.nl/ast/im/gato.jpg";
  popupwin = window.open(document.myurl[m],"mywind","width=250,
    height=250");
}
</script>
</head>

<body>
<p> <a href="#" onMouseover="pop(0); return false;" > Gato </a> </p>
<p> <a href="#" onMouseover="pop(1); return false;" > Perro </a> </p>
<p> <a href="#" onMouseover="pop(2); return false;" > Conejo </a> </p>
</body>
</html>
```

Figura 7-39. Una página Web interactiva que responde al movimiento del ratón.

JavaScript no es la única forma de crear páginas Web altamente interactivas. Otro popular método es a través de **subprogramas** (*applets*). Éstos son pequeños programas de Java que se han compilado en instrucciones de máquina para una computadora virtual llamada **JVM** (**Máquina Virtual de Java**). Los subprogramas pueden incrustarse en páginas HTML (entre <applet> y </applet>) e interpretarse mediante navegadores con capacidad JVM. Debido a que los subprogramas de Java se interpretan en lugar de ejecutarse directamente, el intérprete de Java puede evitar que causen daños. Por lo menos en teoría. En la práctica, los escritores de subprogramas han encontrado un flujo infinito de errores en las bibliotecas de E/S de Java de los cuales aprovecharse.

La respuesta de Microsoft a los subprogramas de Java de Sun fue permitir que las páginas Web contuvieran **controles ActiveX,** que son programas compilados para lenguaje de máquina Pentium y ejecutados en el hardware. Esta característica los hace inmensamente rápidos y más flexibles que los subprogramas interpretados de Java porque pueden hacer cualquier cosa que un programa pueda hacer. Cuando Internet Explorer ve un control ActiveX en una página Web, lo descarga, verifica su identidad y lo ejecuta. Sin embargo, la descarga y ejecución de programas extraños aumenta los problemas de seguridad, lo cual veremos en el capítulo 8.

Puesto que casi todos los navegadores pueden interpretar los programas de Java y JavaScript, un diseñador que desee crear una página Web altamente interactiva puede elegir por lo menos de entre dos técnicas, y si la portabilidad a múltiples plataformas no es importante, también puede elegir ActiveX. Como regla general, los programas de JavaScript son fáciles de escribir, los subprogramas de Java se ejecutan muy rápido y los controles ActiveX se ejecutan todavía más rápido. Además, puesto que todos los navegadores implementan la misma JVM pero no todos implementan la misma versión de JavaScript, los subprogramas de Java son más portables que los programas de JavaScript. Para mayor información sobre JavaScript, hay muchos libros, cada uno con muchas páginas (por lo general, más de 1000). Algunos ejemplos son (Easttom, 2001; Harris, 2001, y McFedries, 2001).

Antes de dejar el tema del contenido dinámico Web, resumamos brevemente lo que hemos visto hasta ahora. Las páginas Web pueden generarse en la marcha mediante varias secuencias de comandos en la máquina servidora. Una vez que el navegador las recibe, las trata como páginas HTML normales y simplemente las despliega. Las secuencias de comandos pueden escribirse en Perl, PHP, JSP o ASP, como se muestra en la figura 7-40.

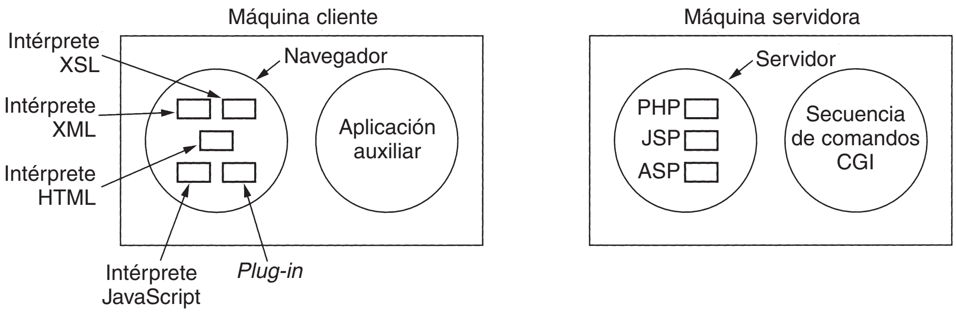


Figura 7-40. Las diversas formas de generar y desplegar contenido.

La generación de contenido dinámico también es posible en el cliente. Es posible escribir en XML las páginas Web y después convertirlas en HTML de acuerdo a un archivo XSL. Los programas de JavaScript pueden realizar cálculos arbitrarios. Por último, es posible utilizar los *plug-ins* y las aplicaciones auxiliares para desplegar contenido en varios formatos.

7.3.4 HTTP—Protocolo de Transferencia de Hipertexto

El protocolo de transferencia utilizado en World Wide Web es **HTTP** (**Protocolo de Transferencia de Hipertexto**). Especifica cuáles mensajes pueden enviar los clientes a los servidores y qué respuestas obtienen. Cada interacción consiste en una solicitud ASCII, seguida por una respuesta tipo MIME del RFC 822. Todos los clientes y servidores deben obedecer este protocolo. Se define en el RFC 2616. En esta sección veremos algunas de sus propiedades más importantes.

Conexiones

La forma común en que un navegador contacta a un servidor es estableciendo una conexión TCP con el puerto 80 de la máquina del servidor, aunque este procedimiento no se requiere formalmente. El valor de utilizar TCP es que ni los navegadores ni los servidores tienen que preocuparse por los mensajes largos, perdidos o duplicados, ni por las confirmaciones de recepción. En la implementación de TCP se manejan todos estos aspectos.

En HTTP 1.0, una vez que se establecía la conexión, se enviaba una solicitud y se obtenía una respuesta. Después se liberaba dicha conexión. Este método era adecuado en un mundo en el que una página Web típica consistía por completo de texto HTML. Sin embargo, años más tarde la página Web promedio contenía una gran cantidad de iconos, imágenes, entre otras cosas, por lo que establecer una conexión TCP para transportar un solo icono era un método muy costoso.

Debido a lo anterior se diseñó HTTP 1.1, que soporta **conexiones persistentes.** Con ellas, es posible establecer una conexión TCP, enviar una solicitud y obtener una respuesta, y después enviar solicitudes adicionales y obtener respuestas adicionales. Al repartir la configuración y terminación de sesiones de TCP en múltiples solicitudes, la sobrecarga resultante de TCP es mucho menor por solicitud. También es posible enviar solicitudes en canalización, es decir, enviar la solicitud 2 antes de que la respuesta a la solicitud 1 haya llegado.

Métodos

Aunque HTTP se diseñó para utilizarlo en Web, se ha hecho intencionalmente más general que lo necesario con miras a las aplicaciones orientadas a objetos futuras. Por esta razón, se soportan otras operaciones, llamadas **métodos,** diferentes a las de solicitar una página Web. Esta generalidad es lo que permite que SOAP exista. Cada solicitud consiste en una o más líneas de texto ASCII, y la primera palabra de la primera línea es el nombre del método solicitado. En la figura 7-41 se listan los métodos integrados. Para acceder a objetos generales, también están disponibles métodos adicionales específicos de objetos. Los nombres son sensibles a mayúsculas y minúsculas, por lo que *GET* es un método válido y *get* no lo es.

El método *GET* solicita al servidor que envíe la página (con lo cual queremos decir objeto, en el caso más general, pero en la práctica se trata normalmente tan sólo de un archivo). La página está codificada de forma adecuada en MIME. La mayoría de las solicitudes a servidores Web son *GET*s. La forma común de *GET* es

```
GET nombredearchivo HTTP/1.1
```

donde *nombredearchivo* es el recurso (archivo) que se va a obtener y 1.1 es la versión del protocolo que se está utilizando.

El método *HEAD* simplemente solicita el encabezado del mensaje, sin la página real. Este método puede utilizarse para obtener la fecha de la última modificación de la página, para colectar información para propósitos de indización o simplemente para proporcionar un URL para validación.

Método	Descripción
GET	Solicita la lectura de una página Web
HEAD	Solicita la lectura del encabezado de una página Web
PUT	Solicita el almacenamiento de una página Web
POST	Inserta algo a un recurso con nombre (por ejemplo, una página Web)
DELETE	Elimina la página Web
TRACE	Repite la solicitud entrante
CONNECT	Reservado para uso futuro
OPTIONS	Consulta ciertas opciones

Figura 7-41. Los métodos de solicitud HTTP integrados.

El método *PUT* es lo inverso a *GET:* en lugar de leer la página, la escribe. Este método posibilita la construcción de una colección de páginas Web en un servidor remoto. El cuerpo de la solicitud contiene la página. Puede codificarse utilizando MIME, en cuyo caso las líneas que siguen al método *PUT* podrían incluir encabezados *Content-Type* y de autenticación, para probar que el invocador tiene permiso de realizar la operación solicitada.

El método *POST* es similar a *PUT*. También transporta un URL, pero en lugar de reemplazar los datos existentes, los nuevos datos se "insertan" en él en algún sentido generalizado. Enviar un mensaje a un grupo de noticias o agregar un archivo a un sistema de boletín de noticias son ejemplos de agregar en este contexto. En la práctica, ni *PUT* ni *POST* se utilizan mucho.

DELETE hace lo que usted espera: elimina la página. Al igual que con *PUT*, la autenticación y el permiso juegan un papel importante aquí. *DELETE* no siempre tendrá éxito, puesto que aunque el servidor HTTP remoto esté dispuesto a eliminar la página, el archivo subyacente puede tener un modo que prohíba al servidor HTTP modificarla o eliminarla.

El método *TRACE* es para la depuración. Indica al servidor que regrese la solicitud. Este método es útil cuando las solicitudes no se están procesando de manera correcta y el cliente desea saber cuál solicitud ha obtenido realmente el servidor.

El método *CONNECT* no se utiliza en la actualidad. Está reservado para uso futuro.

El método *OPTIONS* proporciona una forma para que el cliente consulte al servidor sobre sus propiedades o las de un archivo específico.

Cada solicitud obtiene una respuesta que consiste en una línea de estado, y posiblemente de información adicional (por ejemplo, toda o parte de una página Web). La línea de estado contiene un código de estado de tres dígitos que indica si la solicitud fue atendida, y si no, por qué. El primer dígito se utiliza para dividir las respuestas en cinco grupos mayores, como se muestra en la figura 7-42. En la práctica, los códigos 1xx no se utilizan con frecuencia. Los códigos 2xx significan que la solicitud se manejó de manera exitosa y que se regresa el contenido (si hay alguno). Los códigos 3xx indican al cliente que busque en otro lado, ya sea utilizando un URL diferente o en su propio caché (que se analiza posteriormente). Los códigos 4xx significan que la solicitud

Código	Significado	Ejemplos
1xx	Información	100 = el servidor está de acuerdo en manejar la solicitud del cliente
2xx	Éxito	200 = la solicitud es exitosa; 204 = no hay contenido
3xx	Redirección	301 = página movida; 304 = la página en caché aún es válida
4xx	Error del cliente	403 = página prohibida; 404 = página no encontrada
5xx	Error del servidor	500 = error interno del servidor; 503 = trata más tarde

Figura 7-42. Los grupos de respuesta del código de estado.

falló debido a un error del cliente, por ejemplo una solicitud inválida o una página no existente. Por último, los errores 5xx significan que el servidor tiene un problema, debido a un error en su código o a una sobrecarga temporal.

Encabezados de mensaje

A la línea de solicitud (por ejemplo, la línea con el método *GET*) le pueden seguir líneas adicionales que contienen más información. Éstas se llaman **encabezados de solicitud.** Esta información puede compararse con los parámetros de una llamada a procedimiento. Las respuestas también pueden tener **encabezados de respuesta.** Algunos encabezados pueden utilizarse en cualquier dirección. En la figura 7-43 se muestra una selección de los más importantes.

El encabezado *User-Agent* permite que el cliente informe al servidor sobre su navegador, sistema operativo y otras propiedades. En la figura 7-34 vimos que el servidor tenía mágicamente esta información y podía producirla a solicitud en una secuencia de comandos PHP. El cliente utiliza este encabezado para proporcionarle la información al servidor.

Los cuatro encabezados *Accept* indican al servidor lo que el cliente está dispuesto a aceptar en caso de que tenga un repertorio limitado de lo que es aceptable. El primer encabezado especifica los tipos MIME aceptados (por ejemplo, text/html). El segundo proporciona el conjunto de caracteres (por ejemplo, ISO-8859-5 o Unicode-1-1). El tercero tiene que ver con métodos de compresión (por ejemplo, gzip). El cuarto indica un idioma natural (por ejemplo, español). Si el servidor tiene la opción de páginas, puede utilizar esta información para proporcionar la que el cliente esté buscando. Si es incapaz de satisfacer la solicitud, se regresa un código de error y la solicitud falla.

El encabezado *Host* da nombre al servidor. Este encabezado se toma del URL y es obligatorio. Se utiliza porque algunas direcciones IP pueden proporcionar múltiples nombres DNS y el servidor necesita una forma de indicar a cuál *host* entregarle la solicitud.

El encabezado *Authorization* es necesario para páginas que están protegidas. Por ejemplo, tal vez el cliente debe probar que tiene permiso para ver la página solicitada. Este encabezado se utiliza en estos casos.

Aunque las *cookies* se tratan en el RFC 2109 en lugar de en el RFC 2616, también tienen dos encabezados. Los clientes utilizan el encabezado *Cookie* para regresar al servidor una *cookie* que fue enviada previamente por alguna máquina del dominio del servidor.

Encabezado	Tipo	Contenido
User-Agent	Solicitud	Información acerca del navegador y su plataforma
Accept	Solicitud	El tipo de páginas que el cliente puede manejar
Accept-Charset	Solicitud	Los conjuntos de caracteres que son aceptables para el cliente
Accept-Encoding	Solicitud	Las codificaciones de página que el cliente puede manejar
Accept-Language	Solicitud	Los idiomas naturales que el cliente puede manejar
Host	Solicitud	El nombre DNS del servidor
Authorization	Solicitud	Una lista de las credenciales del cliente
Cookie	Solicitud	Regresa al servidor una cookie establecida previamente
Date	Ambas	Fecha y hora en que se envió el mensaje
Upgrade	Ambas	El protocolo al que el emisor desea cambiar
Server	Respuesta	Información acerca del servidor
Content-Encoding	Respuesta	Cómo se codifica el contenido (por ejemplo, gzip)
Content-Language	Respuesta	El idioma natural utilizado en la página
Content-Length	Respuesta	La longitud de la página en bytes
Content-Type	Respuesta	El tipo MIME de la página
Last-Modified	Respuesta	La fecha y hora de la última vez que cambió la página
Location	Respuesta	Un comando para que el cliente envíe su solicitud a cualquier otro lugar
Accept-Ranges	Respuesta	El servidor aceptará solicitudes de rango de bytes
Set-Cookie	Respuesta	El servidor desea que el cliente guarde una cookie

Figura 7-43. Algunos encabezados de mensaje HTTP.

El encabezado *Date* puede utilizarse en las dos direcciones y contiene la fecha y la hora en la que se envió el mensaje. El encabezado *Upgrade* se utiliza para facilitar la transición a una versión futura (posiblemente incompatible) del protocolo HTTP. Permite que el cliente anuncie lo que puede soportar y que el servidor afirme lo que está utilizando.

Ahora veamos los encabezados que son utilizados exclusivamente por el servidor en respuesta a las solicitudes. El primero, *Server*, permite que el servidor indique quién es y algunas de sus propiedades, si lo desea.

Los siguientes cuatro encabezados, los que comienzan con *Content-*, permiten que el servidor describa propiedades de la página que está enviando.

El encabezado *Last-Modified* indica cuándo se modificó la página por última vez. Este encabezado juega un papel importante en el almacenamiento en caché de la página.

El servidor utiliza el encabezado *Location* para informar al cliente que debe tratar con un URL diferente. Éste puede utilizarse si la página se movió o para permitir que múltiples URLs hagan referencia a la misma página (posiblemente en servidores diferentes). También se utiliza para compañías que tienen una página Web principal en el dominio *com*, pero que redirigen a los clientes a una página nacional o regional con base en su dirección IP o idioma preferido.

Si una página es muy grande, tal vez un cliente pequeño no la quiera toda de una vez. Algunos servidores aceptan solicitudes de rangos de bytes, por lo que la página puede obtenerse en múltiples unidades pequeñas. El encabezado *Accept-Ranges* indica si el servidor está dispuesto a manejar este tipo de solicitud parcial de páginas.

El segundo encabezado de *cookie*, *Set-Cookie*, es la forma en que los servidores envían *cookies* a los clientes. Se espera que el cliente guarde la *cookie* y la regrese al servidor en solicitudes subsiguientes.

Uso de ejemplo de HTTP

Debido a que HTTP es un protocolo ASCII, es muy fácil que una persona en una terminal (a diferencia de un navegador) hable de manera directa con los servidores Web. Todo lo que se necesita es una conexión TCP con el puerto 80 del servidor. Se alienta a los lectores a que prueben este escenario personalmente (de preferencia desde un sistema UNIX, porque algunos otros sistemas no regresan el estado de la conexión). La siguiente secuencia de comandos lo hará:

```
telnet www.ietf.org 80 >log
GET /rfc.html HTTP/1.1
Host: www.ietf.org

close
```

Esta secuencia de comandos inicia una conexión telnet (es decir, TCP) con el puerto 80 en el servidor Web de IETF, *www.ietf.org*. El resultado de la sesión se redirige a un archivo de *registro* para una inspección posterior. A continuación se encuentra el comando *GET* que nombra al archivo y al protocolo. La siguiente línea es el encabezado *Host* obligatorio. La línea en blanco también se requiere. Indica al servidor que ya no hay más encabezados de solicitud. El comando *close* indica al programa que termine la conexión.

El *registro* puede inspeccionarse utilizando cualquier editor. Debe comenzar de manera similar al listado de la figura 7-44, a menos que IETF haya cambiado recientemente.

Las primeras tres líneas se envían a la salida desde el programa telnet, no desde el sitio remoto. La línea que comienza con HTTP/1.1 es la respuesta de IETF mediante la cual indica que está dispuesto a hablar HTTP/1.1 con usted. A continuación se encuentran algunos encabezados y después el contenido. Ya vimos todos los encabezados excepto *ETag,* que es un identificador de página único que se relaciona con el almacenamiento en caché, y *X-Pad,* que es no estándar y que tal vez sea una solución para algún navegador con errores.

7.3.5 Mejoras de desempeño

La popularidad de Web ha sido casi su destrucción. Los servidores, enrutadores y las líneas con frecuencia están sobrecargados. Muchas personas han comenzado a llamar a WWW World Wide Wait. Como consecuencia de estos retardos interminables, los investigadores han desarrollado varias técnicas para mejorar el desempeño. A continuación examinaremos tres de ellas: el almacenamiento en caché, la replicación del servidor y las redes de entrega de contenido.

```
Trying 4.17.168.6...
Connected to www.ietf.org.
Escape character is '^]'.
HTTP/1.1 200 OK
Date: Wed, 08 May 2002 22:54:22 GMT
Server: Apache/1.3.20 (Unix) mod_ssl/2.8.4 OpenSSL/0.9.5a
Last-Modified: Mon, 11 Sep 2000 13:56:29 GMT
ETag: "2a79d-c8b-39bce48d"
Accept-Ranges: bytes
Content-Length: 3211
Content-Type: text/html
X-Pad: avoid browser bug

<html>
<head>
<title>Página IETF RFC</title>

<script language="javascript">
function url() {
var x = document.form1.number.value
if (x.length == 1) {x = "000" + x }
if (x.length == 2) {x = "00" + x }
if (x.length == 3) {x = "0" + x }
document.form1.action = "/rfc/rfc" + x + ".txt"
document.form1.submit
}
</script>

</head>
```

Figura 7-44. El inicio de la salida de *www.ietf.org/rfc.html.*

Almacenamiento en caché

Una forma muy sencilla de mejorar el desempeño es guardar las páginas que han sido solicitadas en caso de que se utilicen nuevamente. Esta técnica es especialmente efectiva con páginas que se visitan mucho, como *www.yahoo.com* y *www.cnn.com*. La técnica de guardar páginas para uso posterior se conoce como **almacenamiento en caché.** El procedimiento común es que algún proceso, llamado **proxy,** mantenga el caché. Para utilizar el almacenamiento en caché, un navegador puede configurarse para que todas las solicitudes de páginas se le hagan a un *proxy* en lugar de al servidor real de la página. Si el *proxy* tiene la página, la regresa de inmediato. De lo contrario, la obtiene del servidor, la agrega al caché para uso posterior y la envía al cliente que la solicitó.

Dos preguntas importantes relacionadas con el almacenamiento en caché son las siguientes:

1. ¿Quién debe realizar el almacenamiento en caché?
2. ¿Por cuánto tiempo deben almacenarse en caché las páginas?

Hay diversas respuestas para la primera pregunta. Las PCs individuales con frecuencia ejecutan los *proxies* a fin de que éstos puedan buscar con rapidez las páginas previamente visitadas. En una LAN de una compañía, el *proxy* con frecuencia es una máquina compartida por todas las máquinas de la LAN, por lo que si un usuario busca cierta página y luego otro usuario de la misma LAN desea la misma página, ésta se puede obtener del caché del *proxy*. Muchos ISPs también ejecutan *proxies*, a fin de que todos sus clientes puedan tener un acceso más rápido. Con frecuencia, todos estos cachés funcionan al mismo tiempo, por lo que las solicitudes primero van al *proxy* local. Si éste falla, consulta al *proxy* de la LAN. Si éste también falla, prueba el *proxy* del ISP. Este último debe tener éxito, ya sea desde su caché, un caché de nivel superior o desde el servidor mismo. Un esquema que involucra el acceso en secuencia de múltiples cachés se conoce como **almacenamiento en caché jerárquico.** En la figura 7-45 se ilustra una posible implementación.

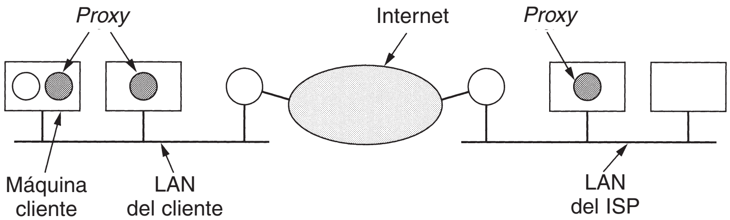


Figura 7-45. Almacenamiento en caché jerárquico con tres *proxies*.

Determinar el tiempo que deben permanecer las páginas en caché es un poco difícil. Algunas páginas no deberían almacenarse en caché. Por ejemplo, una página que contiene los precios de las 50 acciones más activas cambia cada segundo. Si se almacenara en caché, un usuario que obtuviera una copia del caché obtendría datos **obsoletos.** Por otro lado, una vez que el mercado de valores cierre, esa página permanecerá válida por algunas horas o días, hasta que inicie la siguiente sesión del mercado. Por lo tanto, el almacenamiento en caché de una página podría variar con el tiempo.

El elemento clave para determinar cuándo expulsar una página del caché es qué tanta obsolescencia están dispuestos a aceptar los usuarios (debido a que las páginas en caché se almacenan en el disco, la cantidad de espacio consumido por lo general no es un problema). Si un *proxy* elimina páginas con rapidez, raramente regresará una página obsoleta pero tampoco será muy efectivo (es decir, tendrá una tasa baja de coincidencias). Si mantiene las páginas por mucho tiempo, tal vez tendrá una tasa alta de coincidencias pero a expensas de regresar con frecuencia páginas obsoletas.

Hay dos métodos para tratar este problema. El primero utiliza una heurística para adivinar cuánto tiempo se mantendrá cada página. Una común es basar el tiempo de almacenamiento en el encabezado *Last-Modified* (vea la figura 7-43). Si una página se modificó hace una hora, se

mantendrá en caché por una hora. Si se modificó hace un año, obviamente es una página muy estable (por ejemplo, una lista de los dioses de las mitologías griega y romana), por lo que se puede almacenar en caché por un año y esperar razonablemente que no cambie durante dicho tiempo. Si bien esta heurística con frecuencia funciona bien en la práctica, regresa páginas obsoletas de vez en cuando.

El otro método es más costoso, pero elimina la posibilidad de obtener páginas obsoletas debido a que utiliza características especiales del RFC 2616 que tienen que ver con el manejo del caché. Una de las características más útiles es el encabezado de solicitud *If-Modified-Since*, que puede ser enviado por un *proxy* a un servidor. Especifica la página que el *proxy* desea y la fecha en que ésta fue modificada por última vez (a partir del encabezado *Last-Modified*). Si la página no se ha modificado desde entonces, el servidor regresa un mensaje corto *Not Modified* (el código de estado 304 de la figura 7-42), el cual indica al *proxy* que utilice la página en caché. Si ésta ha sido modificada, se regresa la nueva página. Aunque este método siempre requiere un mensaje de solicitud y uno de respuesta, este último será muy corto cuando la entrada en caché aún sea válida.

Estos dos métodos pueden combinarse con facilidad. Para la primera ΔT después de obtener la página, el *proxy* simplemente la regresa a los clientes que la solicitan. Después de que la página ha existido por algún tiempo, el *proxy* utiliza mensajes *If-Modified-Since* para verificar qué tan actualizada está. Elegir ΔT involucra invariablemente algún tipo de heurística, dependiendo de cuándo fue la última modificación de la página.

Las páginas Web con contenido dinámico (por ejemplo, generado por una secuencia de comandos PHP) nunca deben almacenarse en caché debido a que los parámetros pueden ser diferentes la siguiente vez. Para manejar éste y otros casos, hay un mecanismo general para que un servidor instruya a todos los *proxies* de la ruta hacia el cliente que no utilicen otra vez la página actual sin antes verificar si está actualizada. Este mecanismo también puede utilizarse para cualquier página que se espera que cambie rápidamente. En el RFC 2616 también se define una variedad de otros mecanismos de control de caché.

Otro enfoque para mejorar el desempeño es el almacenamiento en caché proactivo. Cuando un *proxy* obtiene una página de un servidor, puede inspeccionarla para ver si tiene hipervínculos. De ser así, puede emitir solicitudes a los servidores correspondientes para precargar el caché con las páginas a las que apuntan dichos hipervínculos, en caso de que éstas se necesiten. Esta técnica puede reducir el tiempo de acceso en las solicitudes subsiguientes, pero también puede inundar las líneas de comunicación con páginas que nunca se solicitan.

Claramente, el almacenamiento en caché de Web es todo menos trivial. Se puede decir mucho más sobre él. De hecho, se han escrito libros completos sobre él, por ejemplo (Rabinovich y Spatscheck, 2002, y Wessels, 2001). Sin embargo, es tiempo de que pasemos al siguiente tema.

Replicación del servidor

El almacenamiento en caché es una técnica en el cliente para mejorar el desempeño, pero también existen técnicas en el servidor. El método más común que los servidores utilizan para mejorar el desempeño es replicar su contenido en múltiples ubicaciones separadas considerablemente. Esta técnica a veces se conoce como **espejeo** (*mirroring*).

Un uso típico del espejeo es para que la página principal de una compañía contenga algunas imágenes e hipervínculos para, digamos, los sitios Web regionales este, oeste, norte y sur de dicha compañía. A continuación el usuario hace clic en el más cercano para acceder a ese servidor. De ahí en adelante, todas las solicitudes van al servidor seleccionado.

Los sitios espejo por lo general son estáticos. La compañía decide dónde colocar los espejos, arregla un servidor en cada región y coloca el contenido en cada ubicación (posiblemente omitiendo los quitanieve del sitio de Miami, y las toallas de playa del sitio de Anchorage). La elección de sitios por lo general permanece estable por meses o años.

Desafortunadamente, Web tiene un fenómeno conocido como **flash crowds** (*aglomeración instantánea*) en el que un sitio Web que era un lugar alejado, desconocido y no visitado de repente se vuelve el centro del universo. Por ejemplo, hasta el 6 de noviembre de 2000, el sitio Web de la Secretaría del Estado de Florida, *www.dos.state.fl.us*, se encontraba tranquilamente proporcionando información sobre las reuniones del gabinete del Estado de Florida e instrucciones de cómo convertirse en un notario en Florida. Pero el 7 de noviembre, cuando la presidencia de los Estados Unidos dependía repentinamente de unos cuantos miles de votos controvertidos en un puñado de condados de Florida, se convirtió en uno de los cinco sitios más visitados del mundo. No es necesario decir que no pudo manejar la carga y casi muere en el intento.

Lo que se necesita es una forma para que un sitio Web que de repente note un incremento masivo en el tráfico, se clone automáticamente a sí mismo en tantas ubicaciones como sea necesario y mantenga funcionando a esos sitios hasta que pase la tormenta, en cuyo momento deberá desactivar muchos o todos ellos. Para tener esta capacidad, un sitio necesita un acuerdo por adelantado con alguna compañía que posea muchos sitios de hospedaje, que indique que puede crear réplicas bajo demanda y que debe pagar por la capacidad que realmente utilice.

Una estrategia aún más flexible es crear réplicas dinámicas por página dependiendo de dónde viene el tráfico. Alguna investigación sobre este tema se reporta en (Pierre y cols., 2001, y Pierre y cols., 2002).

Redes de entrega de contenido

La brillantez del capitalismo es que alguien ha descubierto cómo hacer dinero con World Wide Wait. Funciona como se muestra a continuación. Las compañías llamadas **CDN**s (**redes de entrega de contenido**) hablan con proveedores de contenido (sitios de música, periódicos y otros que desean que su contenido esté disponible rápida y fácilmente) y ofrecen entregar su contenido a los usuarios finales de manera eficiente a cambio de una cuota. Después de que se firma el contrato, el dueño del contenido da a la CDN el contenido de su sitio Web para que lo preprocese (lo que se analizará más adelante) y después lo distribuya.

A continuación la CDN habla con una gran cantidad de ISPs y ofrece pagarles bien a cambio de que le permitan colocar en sus LANs un servidor manejado de manera remota lleno de contenido valioso. Ésta no es sólo una fuente de ingresos, sino que también proporciona a los clientes de los ISPs excelente tiempo de respuesta para obtener el contenido de la CDN, lo que da al ISP una ventaja competitiva sobre otros ISPs que no han tomado dinero gratis de la CDN. Bajo estas condiciones, firmar con una CDN es una decisión que el ISP no necesita pensar. Como

consecuencia, las CDNs más grandes tienen más de 10,000 servidores distribuidos por todo el mundo.

Con el contenido replicado en miles de sitios alrededor del mundo, hay claramente un gran potencial para mejorar el rendimiento. Sin embargo, para que esto funcione, debe haber una forma para redirigir la solicitud del cliente al servidor más cercano de la CDN, de preferencia uno colocado en el ISP del cliente. Además, esta redirección se debe hacer sin modificar el DNS o cualquier otra parte de la infraestructura estándar de Internet. A continuación se da una descripción ligeramente simplificada de cómo lo hace Akamai, la CDN más grande.

Todo el proceso inicia cuando el proveedor de contenido entrega a la CDN su sitio Web. A continuación la CDN ejecuta cada página a través de un preprocesador que reemplaza todos los URLs con URLs modificados. El modelo funcional detrás de esta estrategia es que el sitio Web del proveedor de contenido conste de muchas páginas que sean pequeñas (sólo texto HTML), pero que éstas por lo general tengan vínculos a archivos más grandes, como imágenes, audio y vídeo. Las páginas HTML modificadas se almacenan en el servidor del proveedor de contenido y se obtienen de la forma común; las imágenes, el audio y el vídeo se encuentran en los servidores de la CDN.

Para ver cómo funciona realmente este esquema, considere la página Web de Furry Video que se muestra en la figura 7-46(a). Después de que preprocesa, se transforma en la que se muestra en la figura 7-46(b) y se coloca en el servidor de Furry Video como *www.furryvideo.com/index.html*.

Cuando el usuario teclea el URL *www.furryvideo.com*, el DNS regresa la dirección IP del sitio Web de Furry Video, permitiendo que la página principal (HTML) se obtenga de la forma común. Cuando se hace clic en alguno de los hipervínculos, el navegador pide al DNS que busque *cdn-server.com*, lo cual hace. A continuación, el navegador envía una solicitud HTTP a esta dirección IP, esperando obtener un archivo MPEG.

Eso no sucede porque *cdn-server.com* no alberga ese contenido. En su lugar, es el servidor HTTP falso de la CDN. Éste examina el nombre del archivo y del servidor para averiguar cuál página de cuál proveedor de contenido se necesita. También examina la dirección IP de la solicitud entrante y la busca en su base de datos para determinar en dónde es probable que esté el usuario. Una vez que obtiene esta información determina cuál de los servidores de contenido de la CDN puede dar el mejor servicio al usuario. Esta decisión es difícil porque el que esté más cerca geográficamente puede no ser el que esté más cerca en términos de topología de red, y éste puede estar muy ocupado en ese momento. Después de tomar una decisión, *cdn-server.com* regresa una respuesta con el código de estado 301 y un encabezado *Location* en el que se da el URL del archivo del servidor de contenido de la CDN que esté más cerca del cliente. Para este ejemplo, asumamos que el URL es *www.CDN-0420.com/furryvideo/osos.mpg*. A continuación el navegador procesa este URL de la forma usual para obtener el archivo MPEG real.

La figura 7-47 ilustra los pasos involucrados. El primer paso busca a *www.furryvideo.com* para obtener su dirección IP. Después de eso, la página HTML se puede obtener y desplegar de la forma común. La página contiene tres hipervínculos al *cdn-server* [vea la figura 7-46(b)]. Cuando se hace clic en el primero, su dirección DNS se busca (paso 5) y se regresa (paso 6). Cuando se envía a *cdn-server* un mensaje en el que se solicita *osos.mpg* (paso 7), se le indica al cliente que vaya a *CDN-0420.com* (paso 8). Cuando dicho cliente hace lo que se le indica (paso 9), se le da el archivo del caché del *proxy* (paso 10). La propiedad que hace que todo este mecanismo

```
<html>
<head> <title> Furry Video </title> </head>
<body>
<h1> Lista de productos de Furry Video </h1>
<p> Haga clic abajo para obtener muestras gratis. </p>

<a href="osos.mpg"> Los osos hoy </a> <br>
<a href="conejos.mpg"> Conejos graciosos </a> <br>
<a href="ratones.mpg"> Ratones simpáticos </a> <br>
</body>
</html>
```

(a)

```
<html>
<head> <title> Furry Video </title> </head>
<body>
<h1> Lista de productos de Furry Video </h1>
<p> Haga clic abajo para obtener muestras gratis. </p>

<a href="http://cdn-server.com/furryvideo/osos.mpg"> Los osos hoy </a> <br>
<a href="http://cdn-server.com/furryvideo/conejos.mpg"> Conejos
   graciosos </a> <br>
<a href="http://cdn-server.com/furryvideo/ratones.mpg"> Ratones
   simpáticos </a> <br>
</body>
</html>
```

(b)

Figura 7-46. (a) Página Web original. (b) La misma página después de la transformación.

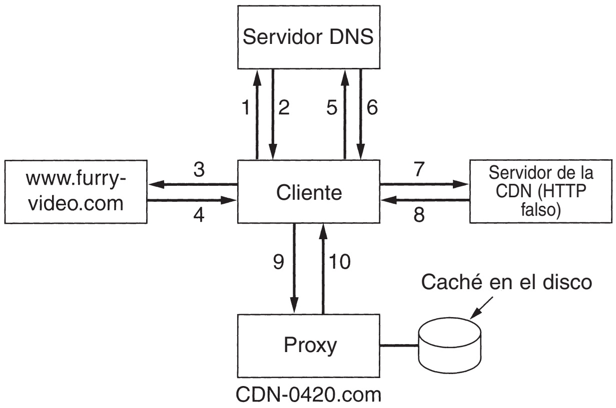
funcione es el paso 8, el servidor HTTP falso que redirige al cliente a un *proxy* de la CDN que está cerca del cliente.

El servidor de la CDN al que se redirige al cliente por lo general es un *proxy* con un caché grande precargado con el contenido más importante. Sin embargo, si alguien pide un archivo que no se encuentra en el caché, se obtiene del servidor real y se coloca en el caché para uso posterior. Al hacer que el servidor de contenido sea un *proxy* en lugar de una réplica, la CDN tiene la capacidad de negociar un tamaño de disco, tiempo de precarga y diversos parámetros de desempeño.

Para obtener mayor información sobre las redes de entrega de contenido, vea (Hull, 2002, y Rabinovich y Spatscheck, 2002).

7.3.6 La Web inalámbrica

Hay un interés considerable en los dispositivos pequeños portables capaces de acceder a Web a través de un enlace inalámbrico. De hecho, ya se han tomado los primeros pasos tentativos en esa dirección. Sin duda habrá muchos cambios en esta área en los años venideros, pero aún vale



1. Se busca www.furryvideo.com
2. Se regresa la dirección IP de Furry
3. Se solicita la página HTML desde Furry
4. Se regresa la página HTML
5. Después de hacer clic, se busca cdn-server.com
6. Se regresa la dirección IP del servidor de la CDN
7. Se pide al servidor de la CDN el archivo osos.mpg
8. Se le indica al cliente que redirija a CDN-0420.com
9. Se solicita el archivo osos.mpg
10. Se regresa el archivo en caché osos.mpg

Figura 7-47. Pasos de la búsqueda de un URL cuando se utiliza una CDN.

la pena examinar algunas de las ideas actuales relacionadas con la Web inalámbrica para ver en dónde estamos y hacia dónde podríamos dirigirnos. Nos enfocaremos en los dos primeros sistemas Web inalámbricos de área amplia: WAP e i-mode.

WAP—Protocolo de Aplicaciones Inalámbricas

Una vez que Internet y los teléfonos móviles se volvieron comunes en todos los lugares, no tomó mucho tiempo para que alguien tuviera la idea de combinarlos en un teléfono móvil con una pantalla integrada para acceder de manera inalámbrica al correo electrónico y a Web. Ese “alguien” en este caso fue un consorcio que inicialmente estaba encabezado por Nokia, Ericsson, Motorola y phone.com (anteriormente Unwired Planet) y que ahora tiene una gran cantidad de miembros. El sistema se llama **WAP** (**Protocolo de Aplicaciones Inalámbricas**).

Un dispositivo WAP puede ser un teléfono móvil mejorado, un PDA o una computadora *notebook* sin ninguna capacidad de voz. La especificación acepta a todos ellos y a otros más. La idea básica es utilizar la infraestructura existente digital inalámbrica. Los usuarios pueden literalmente llamar a una puerta de enlace WAP a través del enlace inalámbrico y enviarle solicitudes de páginas Web. A continuación dicha puerta de enlace verifica su caché para ver si tiene la página solicitada. Si la tiene, la envía; si no la tiene, la obtiene a través de la Internet alámbrica. En esencia, esto significa que WAP 1.0 es un sistema de conmutación de circuitos con un cargo de conexión por minuto relativamente alto. En resumen, a las personas no les gustó acceder a Internet en una pequeña pantalla y pagar por minuto, por lo que WAP fue un fracaso (aunque también habían otros problemas). Sin embargo, parecía que WAP y su competidor, i-mode (que se analizará más adelante), convergían en una tecnología similar, por lo que WAP 2.0 sí podría ser un éxito. Puesto que WAP 1.0 fue el primer intento de una Internet inalámbrica, vale la pena describirla por lo menos brevemente.

WAP es en esencia una pila de protocolos para acceder a Web, pero está optimizada para conexiones de ancho de banda bajo que utilizan dispositivos inalámbricos que tienen una CPU lenta, poca memoria y una pantalla pequeña. Estos requerimientos son muy diferentes de aquellos del escenario de PC de escritorio estándar, lo que provoca algunas diferencias de protocolos. En la figura 7-48 se muestran las capas.

Entorno de Aplicaciones Inalámbricas (WAE)
Protocolo de Sesión Inalámbricas (WSP)
Protocolo de Transacciones Inalámbricas (WTP)
Capa Inalámbrica de Seguridad de Transporte (WTLS)
Protocolo de Datagrama Inalámbrico (WDP)
Capa del portador (GSM, CDMA, D-AMPS, GPRS, etcétera)

Figura 7-48. La pila de protocolos WAP.

La capa inferior soporta todos los sistemas existentes de teléfonos móviles, incluyendo GSM, D-AMPS y CDMA. La tasa de datos de WAP 1.0 es de 9600 bps. Encima de esta capa se encuentra el protocolo de datagramas, **WDP** (**Protocolo de Datagrama Inalámbrico**), que es esencialmente UDP. A continuación se encuentra una capa para seguridad, obviamente necesaria en un sistema inalámbrico. WTLS es un subgrupo de la SSL de Netscape, la cual veremos en el capítulo 8. Arriba de la capa anterior se encuentra la capa de transacciones, la cual maneja de manera confiable o no confiable las solicitudes y respuestas. Esta capa reemplaza a TCP, que no se utiliza a través del enlace de aire por razones de eficiencia. A continuación se encuentra una capa de sesión, que es similar a HTTP/1.1 pero tiene algunas restricciones y extensiones para propósitos de optimización. En la parte superior se encuentra un micronavegador (WAE).

Además del costo, el otro aspecto que sin duda daña la aceptación de WAP es el hecho de que no utiliza HTML. En su lugar, la capa WAE utiliza un lenguaje de marcado llamado **WML** (**Lenguaje de Marcado Inalámbrico**), que es una aplicación de XML. Como consecuencia, en principio, un dispositivo WAP sólo puede acceder aquellas páginas que se han convertido a WML. Sin embargo, debido a que esto restringe el valor de WAP, la arquitectura exige un filtro al vuelo de HTML a WML para incrementar el conjunto de páginas disponibles. Esta arquitectura se ilustra en la figura 7-49.

Con toda justicia, WAP probablemente estaba adelantado a su época. Cuando se inició WAP por primera vez, XML ya era muy conocido fuera del W3C por lo que la prensa anunció su lanzamiento como **WAP NO UTILIZA HTML.** Un encabezado más preciso habría sido: **WAP YA UTILIZA EL NUEVO ESTÁNDAR HTML.** Pero una vez hecho el daño, fue difícil repararlo y WAP 1.0 nunca tuvo popularidad. Analizaremos WAP después de echar un vistazo a su mayor competidor.

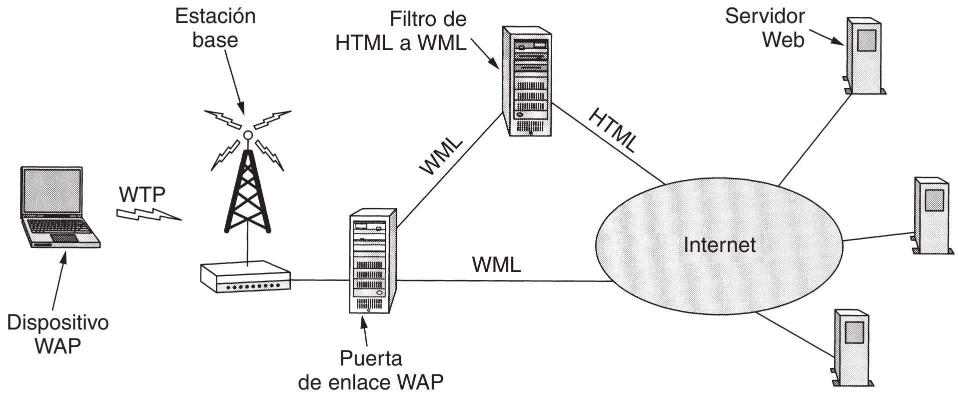


Figura 7-49. La arquitectura de WAP.

I-mode

Mientras un consorcio de múltiples industrias de fabricantes de telecomunicaciones y compañías de computadoras estaba ocupado elaborando con dificultad un estándar abierto utilizando la versión más avanzada disponible de HTML, en Japón se estaban realizando otros desarrollos. Una mujer japonesa, Mari Matsunaga, inventó un método diferente para la Web inalámbrica llamado **i-mode** (***information-mode***)**.** Mari convenció al subsidiario inalámbrico del primer monopolio de telefonía japonesa de que su método era correcto, y en febrero de 1999 NTT DoCoMo (literalmente: Japanese Telephone and Telegraph Company a donde quiera que vaya) lanzó el servicio en Japón. En los siguientes tres años tuvo cerca de 35 millones de suscriptores japoneses, quienes podían acceder aproximadamente 40,000 sitios Web de i-mode especiales. También tenía la admiración de las compañías de telecomunicaciones por su éxito financiero, especialmente debido a que parecía que WAP no iba a ningún lado. A continuación echaremos un vistazo a lo que es i-mode y cómo funciona.

El sistema i-mode tiene tres componentes principales: un nuevo sistema de transmisión, un nuevo microteléfono y un nuevo lenguaje para el diseño de páginas Web. El sistema de transmisión consiste en dos redes separadas: la red de teléfonos móviles de conmutación de circuitos existente (un tanto similar a D-AMPS) y una nueva red de conmutación de paquetes construida específicamente para el servicio i-mode. El modo de voz utiliza la red de conmutación de circuitos y se cobra por minuto de tiempo de conexión. i-mode utiliza la red de conmutación de paquetes y siempre está activo (al igual que ADSL o el cable), por lo que no se cobra una tarifa por tiempo de conexión. En su lugar, se cobra por paquete enviado. En la actualidad no es posible utilizar las dos redes al mismo tiempo.

Los microteléfonos se parecen a los teléfonos móviles, pero además tienen una pantalla pequeña. NTT DoCoMo promociona los dispositivos i-mode como teléfonos móviles mejorados y no como terminales Web inalámbricas, aunque esto es precisamente lo que son. De hecho, probablemente la mayoría de los clientes no están concientes de que están en Internet. Consideran a sus

dispositivos i-mode como teléfonos móviles con servicios mejorados. Al apegarnos a la idea de que el modelo de i-mode es un servicio, los microteléfonos no pueden ser programados por usuarios, aunque contienen el equivalente de una PC de 1995 y probablemente podrían ejecutar Windows 95 o UNIX.

Cuando se hace la conmutación del microteléfono de i-mode, se presenta al usuario una lista de categorías de los servicios aprobados oficialmente. Hay cerca de 1000 servicios divididos en 20 categorías. Cada uno, que en realidad es un pequeño sitio Web i-mode, es ejecutado por una compañía independiente. La categoría principal del menú oficial incluye correo electrónico, noticias, clima, deportes, juegos, compras, mapas, horóscopos, entretenimiento, viajes, guías regionales, sonidos de teléfono, recetas, casinos, sistema bancario doméstico y valores de la bolsa. El servicio está dirigido a adolescentes y a personas en sus 20s, quienes tienden a amar los cosas electrónicas, especialmente si vienen en colores de moda. El simple hecho de que cerca de 40 compañías estén vendiendo sonidos de teléfono significa algo. La aplicación más popular es el correo electrónico, que permite mensajes de hasta 500 bytes y, por lo tanto, se considera como una gran mejora en comparación con SMS (Servicio de Mensajes Cortos) que permite mensajes de hasta 160 bytes. Los juegos también son populares.

También hay cerca de 40,000 sitios Web i-mode, pero tienen que accederse escribiendo su URL, en lugar de seleccionarlos de un menú. En este sentido, la lista oficial es como un portal de Internet que permite que otros sitios Web se accedan haciendo clic en un hipervínculo en lugar de escribir un URL.

NTT DoCoMo controla en forma estricta los servicios oficiales. Para ser aceptado en la lista, un servicio debe cumplir varios criterios publicados. Por ejemplo, un servicio no debe ser una mala influencia para la sociedad, los diccionarios japonés-inglés deben tener suficientes palabras, los servicios que proporcionan sonidos de teléfono deben agregar con frecuencia nuevos sonidos y ningún sitio debe alentar el comportamiento anormal o decir algo malo de NTT DoCoMo (Frengle, 2002). Los 40,000 sitios de Internet pueden hacer lo que quieran.

El modelo de negocios de i-mode es tan diferente del de la Internet convencional que vale la pena explicarlo. La cuota básica de suscripción a i-mode es de algunos dólares mensuales. Puesto que hay un cargo por cada paquete recibido, la suscripción básica incluye un pequeño número de paquetes. De manera alterna, el cliente puede elegir una suscripción con más paquetes gratis, en la que el cargo por paquete desciende bruscamente conforme avanza de 1 MB hasta 10 MB por mes. Si los paquetes gratis se acaban a la mitad del mes, es posible comprar paquetes adicionales en línea.

Para utilizar un servicio, debe suscribirse a él, lo que se logra con sólo hacer clic en él e introduciendo su código PIN. La mayoría de los servicios oficiales cuesta cerca de $1 a $2 mensuales. NTT DoCoMo agrega el cargo al recibo telefónico y da el 91% de tal cargo al proveedor del servicio, y se queda con el 9%. Si un servicio no oficial tiene un millón de clientes, tiene que enviar mensualmente un millón de facturas de (aproximadamente) $1 cada una. Si ese servicio se vuelve oficial, NTT DoCoMo maneja la facturación y sólo transfiere $910,000 mensualmente a la cuenta bancaria del servicio. No tener que manejar la facturación es un gran incentivo para que un proveedor de servicios se vuelva oficial, lo cual genera más ingresos para NTT DoCoMo. Además, el hecho de ser oficial lo coloca en el menú inicial, que hace que su sitio sea más fácil

de encontrar. El recibo telefónico del usuario incluye llamadas telefónicas, cargos por suscripción a i-mode, cargos por suscripción a servicios y paquetes extra.

A pesar de su gran éxito en Japón, aún no es claro si tendrá éxito en Estados Unidos y en Europa. De alguna forma, las circunstancias japonesas son diferentes a las de Occidente. Primero, la mayoría de los clientes potenciales occidentales (por ejemplo, adolescentes, universitarios y personas de negocios) ya tienen en casa una PC con una gran pantalla y, seguramente, con una conexión a Internet con una velocidad de por lo menos 56 kbps, y con frecuencia mucho más. En Japón pocas personas tienen en casa una PC conectada a Internet, en parte debido a la falta de espacio, pero también debido a los exorbitantes cargos de NTT por los servicios locales telefónicos (aproximadamente $700 por instalar una línea y $1.50 la hora por llamadas locales). Para la mayoría de los usuarios, i-mode es la única conexión a Internet.

Segundo, las personas occidentales no están acostumbradas a pagar $1 mensual por acceder al sitio Web de CNN, $1 mensual por acceder al sitio Web de Yahoo, $1 mensual por acceder el sitio Web de Google, etcétera, sin mencionar la cuota por MB descargados. En la actualidad la mayoría de los proveedores de Internet occidentales cobran una cuota mensual sin importar el uso real, principalmente como respuesta a las exigencias del cliente.

Tercero, para muchos japoneses el horario de cuota normal es mientras se trasladan en tren o en el metro. En Europa se trasladan por tren menos personas que en Japón, y en Estados Unidos casi nadie lo hace. Usar i-mode en casa junto a la computadora con monitor de 17 pulgadas, conexión ADSL de 1 Mbps, y todos los megabytes que quiera gratis no tiene mucho sentido. Sin embargo, nadie predijo la inmensa popularidad de los teléfonos móviles, por lo que i-mode aún podría encontrar un lugar en Occidente.

Como mencionamos anteriormente, los microteléfonos de i-mode utilizan la red de conmutación de circuitos existente para voz y una nueva red de conmutación de paquetes para datos. La red de datos se basa en CDMA y transmite paquetes de 128 bytes a 9600 bps. En la figura 7-50 se muestra un diagrama de la red. Los microteléfonos hablan **LTP** (**Protocolo de Transporte de Carga Ligera**) a través del enlace de aire hacia una puerta de enlace de conversión de protocolo. Dicha puerta de enlace tiene una conexión de fibra óptica de banda ancha con el servidor i-mode, que está conectado a todos los servicios. Cuando el usuario selecciona un servicio del menú oficial, la solicitud se envía al servidor i-mode, que almacena en caché la mayoría de las páginas para mejorar el desempeño. Las solicitudes a sitios que no están en el menú oficial ignoran el servidor i-mode y van directamente a través de Internet.

Los microteléfonos actuales tienen CPUs que se ejecutan aproximadamente a 100 MHz, algunos megabytes de memoria ROM (flash ROM), tal vez 1 MB de RAM y una pequeña pantalla integrada. i-mode requiere que la pantalla sea de por lo menos 72×94 píxeles, pero algunos dispositivos de alta calidad pueden tener hasta 120×160 píxeles. Por lo general, las pantallas tienen colores de 8 bits, que permiten 256 colores. Esto no es suficiente para fotografías pero es adecuado para dibujos de líneas y caricaturas sencillas. Puesto que no hay ratón, la navegación en pantalla se realiza con las teclas de flecha.

En la figura 7-51 se muestra la estructura del software. La capa inferior consiste en un sistema operativo en tiempo real sencillo para controlar el hardware. Después se encuentra el módulo para realizar la comunicación de red, que utiliza el protocolo LTP propietario de NTT DoCoMo.

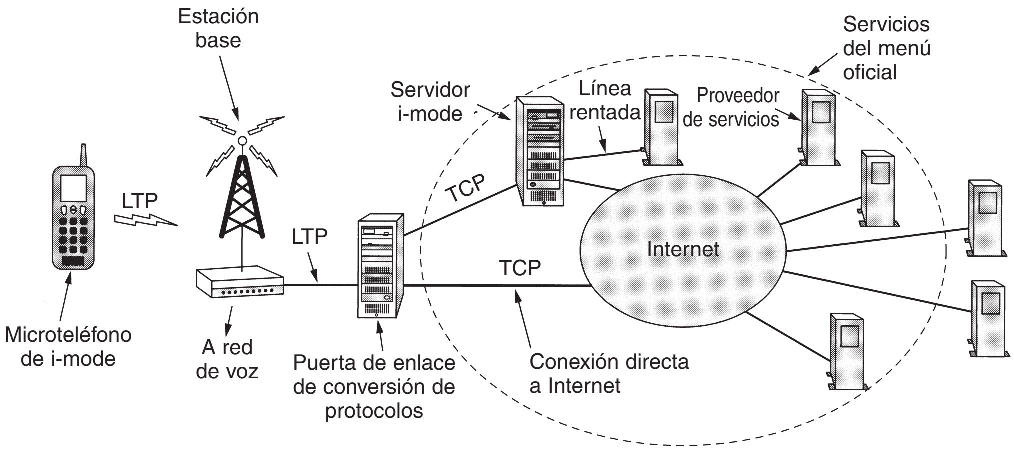


Figura 7-50. Estructura de la red de datos de i-mode que muestra los protocolos de transporte.

Arriba de eso se encuentra un administrador de ventanas sencillo que maneja texto y gráficos simples (archivos GIF). Debido a que las pantallas por lo mucho son de 120 × 160 píxeles, no hay mucho que manejar.

Módulo de interacción con el usuario		
Plug-ins	Intérprete cHTML	Java
Administrador de ventanas sencillas		
Comunicación de red		
Sistema operativo en tiempo real		

Figura 7-51. Estructura del software de i-mode.

La cuarta capa contiene el intérprete de la página Web (es decir, el navegador). i-mode no utiliza el HTML completo, sino un subconjunto de él, llamado **cHTML** (**compact HTML, HTML compacto),** que se basa ligeramente en HTML 1.0. Esta capa también permite aplicaciones auxiliares y *plug-ins*, al igual que lo hacen los navegadores de las PCs. Una aplicación auxiliar estándar es un intérprete de una versión ligeramente modificada de JVM.

En la parte superior se encuentra un módulo de interacción con el usuario, el cual maneja la comunicación con el usuario.

Ahora demos un vistazo más de cerca al cHTML. Como dijimos, es parecido a HTML 1.0, con algunas omisiones y algunas extensiones para utilizarlo en microteléfonos móviles. Se emitió al W3C para su estandarización, pero éste mostró poco interés en él, por lo que es probable que permanezca como un producto propietario.

Se permite la mayoría de las etiquetas básicas HTML, entre ellas <html>, <head>, <title>, <body>, <h*n*>, <center>, <ul>, <ol>, <menu>, <li>,
, <p>, <hr>, <img>, <form> e <input>. Las etiquetas <b> e <i> no están permitidas.

La etiqueta <a> se permite para enlazar páginas, pero con el esquema adicional *tel* para marcar números telefónicos. En un sentido, *tel* es análogo a *mailto*. Cuando se selecciona un hipervínculo que utiliza *mailto*, el navegador despliega un formulario para enviar correo electrónico al destino nombrado en el vínculo. Cuando se selecciona un hipervínculo que utiliza el esquema *tel*, el navegador marca el número telefónico. Por ejemplo, una libreta de direcciones podría tener fotos sencillas de varias personas. Cuando se seleccione una de ellas, el microteléfono llamará a la persona correspondiente. El RFC 2806 analiza los URLs telefónicos.

El navegador cHTML está limitado de otras formas. No soporta JavaScript, tramas, hojas de estilo, colores de fondo o imágenes de fondo. Tampoco soporta imágenes JPEG debido a que tardan mucho en descomprimirse. Los subprogramas de Java están permitidos, pero están (actualmente) limitados a 10 KB debido a la velocidad lenta de transmisión a través del enlace de aire.

Aunque NTT DoCoMo eliminó algunas etiquetas HTML, también agregó otras más. La etiqueta <blink> hace que el texto se encienda y se apague. Aunque parece inconsistente prohibir la etiqueta <b> (sobre la base de que los sitios Web no deben manejar la apariencia) y después agregar la etiqueta <blink>, que se refiere solamente a la apariencia, lo hicieron. <marquee> es otra de las etiquetas nuevas; ésta desplaza el contenido en la pantalla de forma parecida a un teletipo.

Otra característica nueva es el atributo *align* de la etiqueta
. Es necesario debido a que con una pantalla de, por lo general, 6 filas de 16 caracteres, hay un gran riesgo de que las palabras se partan a la mitad, como se muestra en la figura 7-52(a). *Align* ayuda a reducir este problema a fin de obtener algo muy parecido a lo que se muestra en la figura 7-52(b). Es interesante mencionar que los japoneses no tienen el problema de que sus palabras se dividan en diversas líneas. Para el texto kanji, la pantalla se divide en una cuadrícula rectangular de celdas de 9×10 píxeles o 12×12 píxeles, dependiendo de la fuente soportada. Cada celda puede contener exactamente un carácter kanji, que es el equivalente de una palabra en inglés. En japonés están permitidos los saltos de líneas entre palabras.

```
The time has com
e the walrus sai
d to talk of man
y things. Of sho
es and ships and
sealing wax of c
```

(a)

```
The time has
come the walrus
said to talk of
many things. Of
shoes and ships
and sealing wax
```

(b)

Figura 7-52. Lewis Carroll conoce una pantalla de 16×6.

Aunque el lenguaje japonés tiene decenas de miles de kanji, NTT DoCo-Mo inventó otros 166, llamados **emoji,** pictogramas parecidos a las caritas que se muestran en la figura 7-6. Incluyen símbolos para los signos astrológicos, cerveza, hamburguesa, parque de diversiones, cumpleaños, teléfono

móvil, perro, gato, Navidad, corazón roto, beso, estado de ánimo, dormilón y, claro, uno que significa bonito.

Otro nuevo atributo es la capacidad de permitir que los usuarios seleccionen hipervínculos mediante el teclado, una propiedad claramente importante en una computadora sin ratón. En la figura 7-53 se muestra el archivo cHTML que contiene un ejemplo de cómo se utiliza este atributo.

```
<html>
<body>
<h1> Seleccione una opción </h1>
<a href="messages.chtml" accesskey="1"> Verifique el correo de voz </a> <br>
<a href="mail.chtml" accesskey="2"> Verifique el correo electrónico </a> <br>
<a href="games.chtml" accesskey="3"> Ejecute un juego </a>
</body>
</html>
```

Figura 7-53. Un ejemplo de un archivo cHTML.

Auque el cliente está algo limitado, el servidor i-mode es una computadora completamente equipada, con todas las características comunes. Soporta CGI, Perl, PHP, JSP, ASP y todo lo demás que normalmente soportan los servidores Web.

En la figura7-54 se muestra una breve comparación de cómo están implementados WAP e i-mode en los sistemas de primera generación. Aunque algunas de las diferencias parecen pequeñas, por lo general son importantes. Por ejemplo, las personas de 15 años no tienen tarjetas de crédito, por lo que poder comprar productos a través del comercio electrónico y cargarlos al recibo telefónico marcan una gran diferencia en su interés en el sistema.

Para mayor información acerca de i-mode, vea (Frengle, 2002, y Vacca, 2002).

Web inalámbrica de segunda generación

Se suponía que WAP 1.0, basado en estándares internacionales reconocidos, sería una herramienta seria para la gente de negocios en movimiento. Fracasó. I-mode era un juguete electrónico para los adolescentes japoneses que utilizaban un todo propietario. Fue un gran éxito. ¿Qué sucedió a continuación? Cada lado aprendió algo de la primera generación de la Web inalámbrica. El consorcio WAP aprendió que el contenido importa. No tener un gran número de sitios Web que hablan su lenguaje de marcado es fatal. NTT DoCoMo aprendió que un sistema propietario cerrado, estrechamente enlazado con microteléfonos y con la cultura japonesa no es un buen producto de exportación. La conclusión que ambos lados aprendieron es que para convencer a una gran cantidad de sitios Web para que coloquen su contenido en el formato de usted, es necesario tener un lenguaje de marcado estable y abierto que sea aceptado de manera universal. Las guerras de los formatos no son buenas para los negocios.

Ambos servicios están cerca de entrar a la tecnología de la segunda generación de la Web inalámbrica. WAP 2.0 existió primero, por lo que lo utilizaremos para nuestro ejemplo. WAP 1.0

Característica	WAP	I-mode
Qué es	Pila de protocolos	Servicio
Dispositivo	Microteléfono, PDA, notebook	Microteléfono
Acceso	Marcado telefónico	Siempre activo
Red subyacente	Conmutación de circuitos	Dos: circuitos + paquetes
Tasa de datos	9600 bps	9600 bps
Pantalla	Monocroma	A color
Lenguaje de marcado	WML (aplicación XML)	cHTML
Lenguaje de secuencia de comandos	WMLscript	Ninguno
Cargos por uso	Por minuto	Por paquete
Pago por compras	Tarjeta de crédito	Recibo telefónico
Pictogramas	No	Sí
Estandarización	Estándar abierto del foro WAP	Propietario NTT DoCoMo
En dónde se utiliza	Europa, Japón	Japón
Usuario típico	Hombre de negocios	Personas jóvenes

Figura 7-54. Comparación entre WAP de primera generación e i-mode.

obtuvo algunas cosas correctas, y éstas aún funcionan. Por ejemplo, WAP se puede transportar en una variedad de redes. La primera generación utilizó las redes de conmutación de circuitos, pero las redes de conmutación de paquetes siempre fueron una opción y aún lo siguen siendo. Es posible que los sistemas de segunda generación utilicen la conmutación de paquetes, por ejemplo, GPRS. Además, WAP estaba destinado inicialmente para soportar una amplia variedad de dispositivos, desde teléfonos móviles hasta poderosas computadoras, y todavía lo está.

WAP 2.0 también tiene algunas nuevas características. Las más significativas son:

1. Modelo *push* (de actualización automática) y modelo *pull* (de recepción automática).
2. Soporte para integrar la telefonía en las aplicaciones.
3. Mensajería multimedia.
4. Inclusión de 264 pictogramas.
5. Interacción con un dispositivo de almacenamiento.
6. Soporte en el navegador para *plug-ins*.

El modelo *pull* es bien conocido: el cliente solicita una página y la obtiene. El modelo *push* soporta el arribo de datos sin que se le solicite, como una retroalimentación continua de la información de la bolsa o alertas de tráfico.

La voz y los datos están comenzando a combinarse, y WAP 2.0 los soporta en una variedad de formas. Vimos un ejemplo de esto anteriormente con la capacidad de i-mode de llamar a un

número telefónico mediante un hipervínculo de icono o fragmento de texto de la pantalla. Además del correo electrónico y la telefonía, se soporta la mensajería multimedia.

La gran popularidad del emoji de i-mode estimuló al consorcio WAP a inventar 264 de sus propios emoji. Las categorías incluyen animales, aplicaciones, vestido, estados de ánimo, comida, cuerpo humano, género, mapas, música, plantas, deportes, fechas, herramientas, vehículos, armas y clima. Es interesante que el estándar sólo nombra cada pictograma; no proporciona el mapa de bits real, probablemente por miedo a que la representación de "adormilado" o "abrazo" de algunas culturas podría ser insultante para otras. I-mode no tuvo ese problema debido a que estaba destinado para un solo país.

El hecho de proveer una interfaz para almacenamiento no significa que cada teléfono WAP 2.0 tendrá un disco duro grande. La memoria ROM también es un dispositivo de almacenamiento. Una cámara inalámbrica con capacidad WAP podría utilizar la memoria ROM para almacenamiento temporal de imágenes antes de emitir la mejor imagen a Internet.

Por último, los *plug-ins* pueden extender las capacidades del navegador. También se proporciona un lenguaje de secuencias de comandos.

En WAP 2.0 también hay diversas diferencias técnicas. Las dos más significativas tienen que ver con la pila de protocolos y con el lenguaje de marcado. WAP 2.0 continúa soportando la antigua pila de protocolos de la figura 7-48, pero también soporta la pila estándar de Internet con TCP y HTTP/1.1. Sin embargo, se realizaron cuatro pequeños (pero compatibles) cambios (para simplificar el código) a TCP: (1) uso de una ventana fija de 64 KB, (2) inicio rápido, (3) una MTU máxima de 1500 byte y (4) un algoritmo de retransmisión ligeramente diferente. TLS es el protocolo de seguridad de la capa de transporte estandarizado por IETF; lo examinaremos en el capítulo 8. Muchos dispositivos iniciales probablemente contendrán ambas pilas, como se muestra en la figura 7-55.

XHTML	
WSP	HTTP
WTP	TLS
WTLS	TCP
WDP	IP
Capa del portador	Capa del portador
Pila de protocolos de WAP 1.0	Pila de protocolos de WAP 2.0

Figura 7-55. WAP 2.0 soporta dos pilas de protocolos.

La otra diferencia técnica con respecto de WAP 1.0 es el lenguaje de marcado. WAP 2.0 soporta XHTML Basic, que está diseñado para dispositivos inalámbricos pequeños. Debido a que NTT DoCoMo también está de acuerdo en soportar este subconjunto, los diseñadores de sitios Web pueden utilizar este formato y saber que sus páginas funcionarán en la Internet fija y en todos los dispositivos inalámbricos. Estas decisiones terminarán con las guerras de formatos de lenguaje de marcado que impiden el crecimiento de la industria de la Web inalámbrica.

Tal vez sean necesarias unas palabras acerca de XHTML Basic. Está diseñado para teléfonos móviles, televisiones, PDAs, máquinas vendedoras (refrescos, golosinas), localizadores, carros, máquinas tragamonedas y de juego e, incluso, relojes. Por esta razón, no soporta hojas de estilo, secuencias de comandos o tramas, pero sí soporta la mayoría de las etiquetas estándar. Están agrupadas en 11 módulos. Algunos son requeridos; algunos son opcionales. Todos están en XML. En la figura 7-56 se listan los módulos y algunas etiquetas de ejemplo. No explicaremos todas las etiquetas de ejemplo, pero puede encontrar mayor información en *www.w3.org*.

Módulo	¿Requerido?	Función	Etiquetas de ejemplo
Estructura	Sí	Estructura de documentos	body, head, html, title
Texto	Sí	Información	br, code, dfn, em, h*n*, kbd, p, strong
Hipertexto	Sí	Hipervínculos	a
Lista	Sí	Listas de elementos	dl, dt, dd, ol, ul, li
Formularios	No	Formularios de relleno	form, input, label, option, textarea
Tablas	No	Tablas rectangulares	caption, table, td, th, tr
Imagen	No	Imágenes	img
Objeto	No	Subprogramas, mapas, etcétera	object, param
Metainformación	No	Información extra	meta
Vínculo	No	Similar a <a>	link
Base	No	Punto de inicio de URL	base

Figura 7-56. Los módulos y etiquetas de XHTML Basic.

A pesar del acuerdo del uso de XHTML Basic, una amenaza asecha a WAP e i-mode: 802.11. Se supone que la Web inalámbrica de segunda generación se debe ejecutar a 384 kbps, una velocidad mucho mayor que los 9600 bps de la primera generación, pero mucho menor que los 11 Mbps o 54 Mbps ofrecidos por 802.11. Por supuesto, 802.11 no está en todos lados, pero conforme más restaurantes, hoteles, tiendas, compañías, aeropuertos, estaciones de autobús, museos, universidades, hospitales y otras organizaciones decidan instalar estaciones base para sus empleados y clientes, tal vez haya mayor cobertura en las áreas urbanas y las personas estén dispuestas a caminar unas cuadras para sentarse en un restaurante de comida rápida con capacidad de 802.11 para tomarse una tasa de café y enviar correo electrónico. Tal vez los negocios coloquen de manera rutinaria logos de 802.11 junto a los que muestran las tarjetas de crédito que dichos negocios aceptan, y por la misma razón: para atraer a los clientes. Los mapas de la ciudad (descargables, naturalmente) podrían mostrar con color verde las áreas cubiertas y en color rojo las áreas sin cobertura, de manera que las personas puedan vagar de estación base a estación base, al igual que los nómadas se trasladan de oasis a oasis en el desierto.

Aunque los restaurantes de comida rápida puedan instalar rápidamente estaciones base 802.11, tal vez los granjeros no puedan hacerlo, por lo que la cobertura será desigual y limitada a las áreas del centro de la ciudad, debido al rango limitado de 802.11 (unos cuantos cientos de metros por lo mucho). Esto podría llevar a dispositivos inalámbricos de modo dual que utilicen 802.11 si pueden captar una señal y regresar a WAP si no pueden hacerlo.

7.4 MULTIMEDIA

La Web inalámbrica es un desarrollo nuevo y excitante, pero no es el único. Para muchas personas, la multimedia es el Santo Grial de las redes. Cuando se menciona la palabra, los *propeller heads* y las demandas legales comienzan a babear como si vieran un gran banquete. El primero ve retos técnicos inmensos para proporcionar a cada casa vídeo (interactivo) bajo demanda. El último ve en ello ganancias inmensas. Debido a que la multimedia requiere un alto ancho de banda, hacer que funcione en conexiones fijas es difícil. Incluso el vídeo de calidad VHS a través de sistemas inalámbricos está a algunos años de distancia, por lo que nos enfocaremos en los sistemas cableados.

Literalmente, multimedia son dos o más medios. Si el editor de este libro quisiera unirse a la euforia actual por la multimedia, podría anunciar que el libro usa tecnología multimedia. A fin de cuentas, contiene dos medios: texto y gráficos (las figuras). No obstante, cuando la mayoría de la gente habla de multimedia, por lo general se refiere a la combinación de dos o más **medios continuos,** es decir, medios que tienen que ejecutarse durante cierto intervalo de tiempo bien definido, generalmente con alguna interacción con el usuario. En la práctica, por lo común los dos medios son audio y vídeo, es decir, sonido más imágenes en movimiento.

Sin embargo, muchas personas con frecuencia también se refieren al audio puro, como la telefonía de Internet o la radio en Internet, como multimedia, pero no lo es. Realmente, un mejor término es **medios de flujo continuo,** pero también consideraremos el audio en tiempo real como multimedia. En las siguientes secciones analizaremos la forma en que las computadoras procesan el audio y el vídeo, cómo están comprimidos y algunas aplicaciones de red de estas tecnologías. Para un análisis (de tres volúmenes) sobre la multimedia en red, vea (Steinmetz y Nahrstedt, 2002; Steinmetz y Nahrstedt, 2003a, y Steinmetz y Nahrstedt, 2003b).

7.4.1 Introducción al audio digital

Una onda de audio (sonido) es una onda acústica (de presión) de una dimensión. Cuando una onda acústica entra en el oído, el tímpano vibra, causando que los pequeños huesos del oído interno vibren con él, enviando pulsos nerviosos al cerebro. El escucha percibe estos pulsos como sonido. De manera parecida, cuando una onda acústica incide en un micrófono, éste genera una señal eléctrica, que representa la amplitud del sonido como una función del tiempo. La representación, procesamiento, almacenamiento y transmisión de tales señales de audio es una parte principal del estudio de los sistemas multimedia.

La gama de frecuencias perceptibles por el oído humano va de 20 Hz a 20,000 Hz, aunque algunos animales, principalmente los perros, pueden escuchar frecuencias más altas. El oído escucha de manera logarítmica, por lo que la relación entre dos sonidos de amplitudes A y B se expresa convencionalmente en **dB** (**decibeles**) de acuerdo con la fórmula

$$\text{dB} = 10 \log_{10} (A/B)$$

Si definimos como 0 dB el límite inferior de la audibilidad (una presión de unas 0.0003 dinas/cm^2) para una onda senoidal de 1 kHz, una conversación ordinaria es de unos 50 dB y el umbral del dolor es de 120 dB, lo que representa una gama dinámica de un factor de un millón.

El oído es sorprendentemente sensible a variaciones de sonido que duran apenas unos milisegundos. En cambio, el ojo no nota cambios en el nivel de luz que duran unos cuantos milisegundos. El resultado de esta observación es que fluctuaciones de apenas unos cuantos milisegundos durante una transmisión multimedia afectan la calidad del sonido percibido más que a la calidad de la imagen percibida.

Las ondas de audio pueden convertirse a una forma digital mediante un **ADC** (**convertidor analógico a digital**). Un ADC toma un voltaje eléctrico como entrada y genera un número binario como salida. En la figura 7-57(a) se muestra un ejemplo de onda senoidal. Para representar esta señal de manera digital, simplemente la muestreamos cada ΔT segundos, como lo muestra la altura de las barras de la figura 7-57(b). Si una onda de sonido no es una onda senoidal pura, sino una superposición de ondas senoidales en las que la componente de más alta frecuencia es *f,* entonces el teorema de Nyquist (vea el capítulo 2) establece que es suficiente tomar muestras a una frecuencia 2*f*. Muestrear a una frecuencia mayor no tiene ningún valor, porque no están presentes las frecuencias mayores que serían detectadas por dicho muestreo.

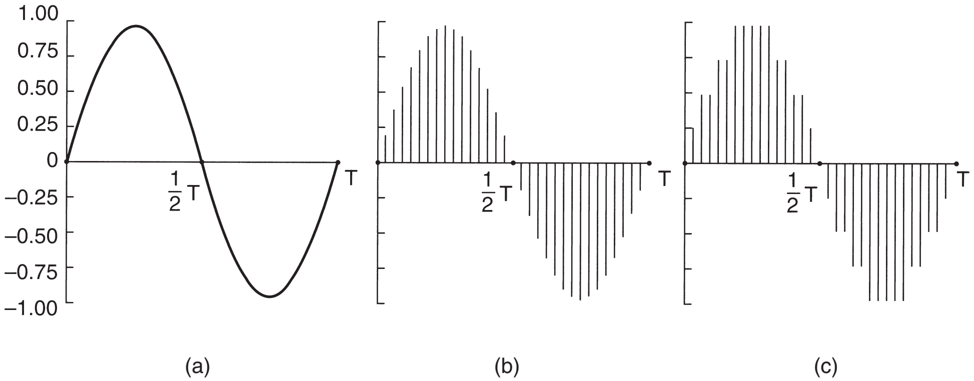


Figura 7-57. (a) Onda senoidal. (b) Muestreo de la onda senoidal. (c) Cuantización de las muestras a 4 bits.

Las muestras digitales nunca son exactas. Las muestras de la figura 7-57(c) sólo permiten nueve valores, de −1.00 a + 1.00 en incrementos de 0.25. Una muestra de 8 bits permitirá 256 valores diferentes. Una muestra de 16 bits permitirá 65,536 valores diferentes. El error introducido por la cantidad finita de bits por muestra se llama **ruido de cuantización.** Si éste es demasiado grande, el oído lo detecta.

Dos ejemplos bien conocidos de sonido muestreado son el teléfono y los discos compactos de audio. La modulación de código de pulso, como la usada en el sistema telefónico, emplea muestras de 8 bits, 8000 veces por segundo. En Norteamérica y Japón, 7 bits son para datos y 1 para control; en Europa, los 8 bits son para datos. Este sistema da una tasa de datos de 56,000 bps o 64,000 bps. Con sólo 8000 muestras/seg, las frecuencias por arriba de 4 kHz se pierden.

Los CDs de audio son digitales, con una tasa de muestreo de 44,100 muestras/seg, suficientes para capturar frecuencias de hasta 22,050 Hz, lo que es bueno para la gente, malo para los perros. Cada una de las muestras tiene 16 bits, y es lineal dentro de la gama de amplitudes. Observe que las muestras de 16 bits permiten sólo 65,536 valores diferentes, aunque la gama dinámica del oído es de aproximadamente 1 millón si se mide en pasos del tamaño del sonido audible más pequeño. Por lo tanto, el uso de sólo 16 bits por muestra genera ruido de cuantización (aunque no se cubre la gama dinámica completa; se supone que los CDs no deben lastimar). Con 44,100 muestras/seg de 16 bits cada una, un CD de audio necesita un ancho de banda de 705.6 kbps para monofónico y 1.411 Mbps para estéreo. Si bien esto es menos de lo que necesita el vídeo (vea más adelante), aun así se requiere un canal T1 completo para transmitir en tiempo real sonido estéreo de calidad CD.

Las computadoras pueden procesar con facilidad mediante software el sonido digital. Existen docenas de programas para que las computadoras personales permitan que los usuarios graben, desplieguen, editen, mezclen y almacenen ondas de sonido de múltiples fuentes. En la actualidad, casi toda la grabación y edición profesional de sonido es digital.

La música, por supuesto, es simplemente un caso especial del audio general, pero es importante. Otro caso especial muy importante es la voz. La voz humana tiende a estar en el rango de 600 a 6000 Hz. La voz se compone de vocales y consonantes, las cuales tienen propiedades diferentes. Las vocales se producen cuando el tracto vocal está libre, produciendo resonancias cuya frecuencia fundamental depende del tamaño y de la forma del sistema vocal y de la posición de la lengua y mandíbula de quien habla. Estos sonidos son casi periódicos en intervalos de aproximadamente 30 mseg. Las consonantes se producen cuando el tracto vocal está bloqueado parcialmente. Estos sonidos son menos regulares que las vocales.

Algunos sistemas de transmisión y generación de voz utilizan modelos del sistema vocal para reducir la voz a unos cuantos parámetros (por ejemplo, los tamaños y las formas de diversas cavidades), en lugar de simplemente muestrear la forma de onda de la voz. Sin embargo, la forma en que funcionan estos codificadores de voz está más allá del alcance de este libro.

7.4.2 Compresión de audio

El audio con calidad de CD requiere un ancho de banda de transmisión de 1.411 Mbps, como acabamos de ver. Claramente, la compresión sustancial se necesita para hacer que la transmisión a través de Internet sea práctica. Por esta razón, se han desarrollado varios algoritmos de compresión de audio. Tal vez el más popular es el audio MPEG, que tiene tres capas (variantes), de las cuales **MP3** (**capa de audio 3 de MPEG**) es la más poderosa y mejor conocida. En Internet hay cantidades considerables de música en formato MP3, no todos legales, lo que ha resultado en varias demandas de los artistas y propietarios de derechos de autor. MP3 pertenece a la porción de audio del estándar de compresión de vídeo de MPEG. Más adelante en este capítulo analizaremos la compresión de vídeo; por ahora veremos la compresión de audio.

La compresión de audio se puede realizar de una de dos formas. En la **codificación de forma de onda** la señal se transforma de manera matemática en sus componentes de frecuencia mediante

una transformación de Fourier. La figura 2-1(a) muestra una función de ejemplo de tiempo y sus amplitudes de Fourier. Por lo tanto, la amplitud de cada componente se codifica en una forma mínima. El objetivo es reproducir la forma de onda de manera precisa en el otro extremo utilizando los menos bits posibles.

La otra forma, **codificación perceptual,** aprovecha ciertas fallas del sistema auditivo humano para codificar una señal a fin de que suene de la misma forma para un escucha, aunque dicha señal luzca de manera diferente en un osciloscopio. La codificación perceptual se basa en la ciencia de **psicoacústica** —cómo perciben las personas un sonido. MP3 se basa en la codificación perceptual.

La propiedad clave de la codificación perceptual es que algunos sonidos pueden **enmascarar** otros sonidos. Imagine que está difundiendo un concierto de flauta en vivo en un día caluroso de verano. De repente, un grupo de trabajadores que está cerca enciende sus martillos perforadores y comienza a romper la calle. Ya nadie puede escuchar la flauta. Sus sonidos han sido enmascarados por los de los martillos perforadores. Para propósitos de transmisión, ahora es suficiente con codificar sólo la banda de frecuencia utilizada por los martillos perforadores, pues de cualquier forma las personas no pueden escuchar la flauta. A esto se le conoce como **enmascaramiento de frecuencia** —la capacidad que tiene un sonido fuerte en una banda de frecuencia de ocultar un sonido más suave en otra banda de frecuencia, el cual podría ser audible si el sonido fuerte no estuviera presente. De hecho, incluso después de que los martillos perforadores pararan, la flauta no se escucharía por un periodo corto debido a que el oído reduce su ganancia cuando los martillos comienzan y toma un tiempo finito para aumentarlo nuevamente. Este efecto se conoce como **enmascaramiento temporal.**

Para hacer que estos efectos sean más cuantitativos, imagine el experimento 1. Una persona en un salón silencioso se pone unos audífonos que están conectados a la tarjeta de sonido de una computadora. Ésta genera una onda senoidal pura a 100 Hz, pero incrementa la potencia de manera gradual. Se le indica a la persona que pulse una tecla cuando escuche el tono. La computadora graba el nivel de potencia actual y después repite el experimento a 200 Hz, 300 Hz y demás frecuencias hasta el límite del oído humano. Cuando se calcula un promedio a partir de varias personas, un gráfico de registro a registro de cuánta potencia se necesita para que un tono sea audible luce como el que se muestra en la figura 7-58(a). Una consecuencia directa de esta curva es que nunca es necesario codificar ninguna frecuencia cuya potencia esté por debajo del umbral de audibilidad. Por ejemplo, si la potencia de 100 Hz fuera 20 dB en la figura 7-58(a), se podría omitir de la salida sin ninguna pérdida perceptible de calidad debido a que 20 dB a 100 Hz están debajo del nivel de audibilidad.

Ahora considere el experimento 2. La computadora realiza otra vez el experimento 1, pero esta vez con una onda senoidal de amplitud constante a, digamos, 150 Hz, superimpuesta en la frecuencia de prueba. Lo que descubrimos es que se incrementa el umbral de audibilidad para las frecuencias que están cerca de 150 Hz, como se muestra en la figura 7-58(b).

La consecuencia de esta nueva observación es que al mantener un registro de cuáles señales están siendo enmascaradas por señales más poderosas de bandas de frecuencia cercanas, podemos omitir más y más frecuencias en la señal codificada, lo que ahorra bits. En la figura 7-58, la señal a 125 Hz se puede omitir por completo de la salida y nadie notará la diferencia. Aunque una

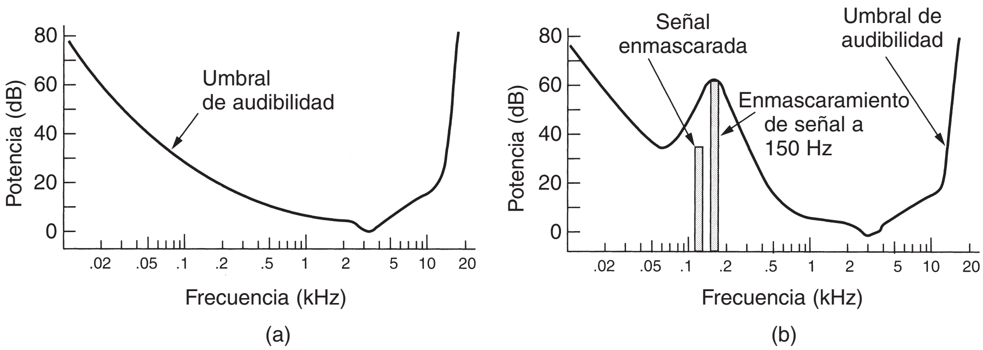


Figura 7-58. (a) Umbral de audibilidad como una función de frecuencia. (b) Efecto de enmascaramiento.

señal poderosa se detenga en alguna banda de frecuencia, si conocemos sus propiedades de enmascaramiento temporal, podemos continuar omitiendo las frecuencias enmascaradas por algún intervalo de tiempo mientras el oído se recupera. La esencia de MP3 es transformar mediante un análisis de Fourier el sonido para obtener la potencia de cada frecuencia y después transmitir sólo las frecuencias no enmascaradas, codificando éstas con los menos bits posibles.

Con esta información como base, ahora podemos ver cómo se realiza la codificación. La compresión de audio se realiza muestreando la forma de onda a 32, 44.1 o 48 kHz. El muestreo puede realizarse en uno de dos canales, en cualquiera de cuatro configuraciones:

1. Monofónica (un solo flujo de entrada).
2. Monofónica dual (por ejemplo, una pista sonora en inglés y una en japonés).
3. Estéreo separado (cada canal se comprime por separado).
4. Estéreo unido (redundancia intercanal completamente explotada).

Primero se elige la tasa de bits de salida. MP3 puede comprimir un CD de estéreo de rock and roll a 96 kbps con una pérdida de calidad apenas perceptible, incluso para los fanáticos del rock and roll que no tienen pérdida del oído. Para un concierto de piano, se necesitan por lo menos 128 kbps. Esto difiere porque la relación señal a ruido para el rock and roll es más alta que la de un concierto de piano (en el sentido de la ingeniería). También es posible elegir tasas de salida más bajas y aceptar alguna pérdida en la calidad.

Después, las muestras se procesan en grupos de 1152 (aproximadamente 26 mseg). Cada grupo primero se pasa a través de 32 filtros digitales para obtener 32 bandas de frecuencia. Al mismo tiempo, la entrada se coloca en un modelo psicoacústico para determinar las frecuencias enmascaradas. A continuación, cada una de las 32 bandas de frecuencia se transforman aún más para proporcionar una resolución espectral más fina.

En la siguiente fase, los bits disponibles se dividen entre las bandas; la mayoría de los bits se asignan a las bandas con la mayor potencia espectral no enmascarada, a las bandas no enmascaradas con menos potencia espectral se les asignan muy pocos bits y a las bandas enmascaradas no se les

asignan bits. Por último, los bits se codifican mediante la codificación de Huffman, que asigna códigos cortos a números que aparecen frecuentemente, y códigos largos a aquellos que no ocurren con frecuencia.

En realidad hay mucho más que decir. También se utilizan varias técnicas para la reducción del ruido, el suavizado y la explotación de la redundancia de intercanal, si es posible, pero éstas están más allá del alcance de este libro. Una introducción matemática a este proceso se proporciona en (Pan, 1995).

7.4.3 Audio de flujo continuo

Ahora movámonos de la tecnología del audio digital a tres de sus aplicaciones de red. La primera es el audio de flujo continuo, es decir, escuchar el sonido a través de Internet. A esto también se le conoce como música bajo demanda. En las siguientes dos veremos la radio en Internet y la voz sobre IP, respectivamente.

Internet está lleno de sitios Web de música, muchos de los cuales listan títulos de canciones en los que los usuarios pueden hacer clic para reproducir esas canciones. Algunos de éstos son sitios gratuitos (por ejemplo, las nuevas bandas que buscan publicidad); otros requieren un pago a cambio de la música, aunque éstos con frecuencia también ofrecen algunas muestras gratis (por ejemplo, los primeros 15 seg de una canción). En la figura 7-59 se muestra la forma más directa para hacer que se reproduzca la música.

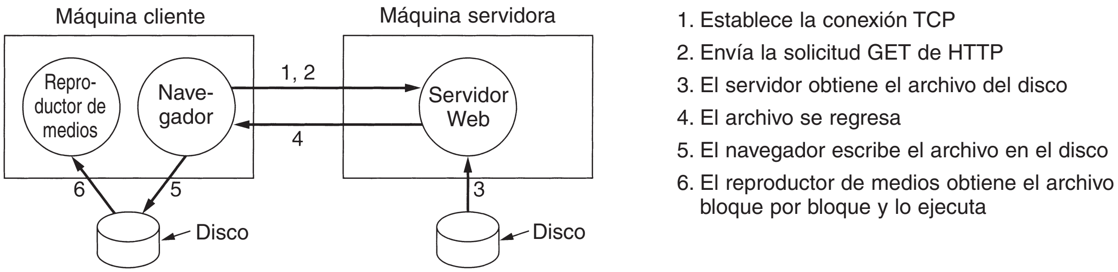


Figura 7-59. Una forma directa de implementar en una página Web música en la que se pueda hacer clic.

El proceso inicia cuando el usuario hace clic en una canción. A continuación el navegador entra en acción. El paso 1 consiste en que éste establezca una conexión TCP con el servidor Web con el que la canción está vinculada. El paso 2 consiste en enviar una solicitud *GET* en HTTP para pedir la canción. A continuación (pasos 3 y 4), el servidor obtiene la canción (que es simplemente un archivo en MP3 o en algún otro formato) del disco y la regresa al navegador. Si el archivo es más grande que la memoria del servidor, tal vez obtenga y envíe la música un bloque a la vez.

El navegador investiga, mediante el tipo MIME, por ejemplo, *audio/mp3* (o la extensión de archivo), cómo se supone que debe desplegar el archivo. Por lo general, habrá una aplicación auxiliar, como RealOne Player, el Reproductor de Windows Media o Winamp, asociado con este tipo de archivos. Debido a que la forma usual de que el navegador se comunique con una aplicación

auxiliar es escribir el contenido en un archivo de trabajo, primero guardará en el disco todo el archivo de música como un archivo de trabajo (paso 5). Después iniciará el reproductor de medios y pasará el nombre del archivo de trabajo. En el paso 6, el reproductor de medios comienza a obtener y a reproducir la música bloque por bloque.

Al principio, este enfoque es correcto y reproducirá la música. El único problema es que la canción completa debe transmitirse a través de la red antes de que comience la música. Si la canción mide 4 MB (un tamaño típico para una canción MP3) y el módem es de 56 kbps, el usuario obtendrá casi 10 minutos de silencio mientras la canción se descarga. No a todos los amantes de la música les gusta esta idea. Especialmente debido a que la siguiente canción también iniciará después de 10 minutos de descarga, y así sucesivamente.

Para resolver este problema sin cambiar la forma en que funciona el navegador, los sitios de música han adoptado el siguiente esquema. El archivo vinculado al título de la canción no es el archivo de música real. En su lugar, es lo que se llama un **metaarchivo,** que es un archivo muy pequeño que sólo nombra a la música. Un metaarchivo típico podría ser una sola línea de texto ASCII y podría lucir como lo siguiente:

```
rtsp://joes-audio-server/song-0025.mp3
```

Cuando el navegador obtiene el archivo de una línea, lo escribe en el disco en un archivo de trabajo, inicia el reproductor de medios como una aplicación auxiliar, y le entrega el nombre del archivo de trabajo, como es usual. A continuación el reproductor de medios lee dicho archivo y ve que contiene un URL. Enseguida contacta al servidor *joes-audio-server* y le pide la canción. Observe que el navegador ya no está en el ciclo.

En muchos casos, el servidor nombrado en el metaarchivo no es el mismo que el servidor Web. De hecho, por lo general ni siquiera es un servidor HTTP, sino un servidor de medios especializado. En este ejemplo, el servidor de medios utiliza **RTSP** (**Protocolo de Transmisión en Tiempo Real**)**,** como se indica en el nombre de esquema *rtsp*. Éste se describe en el RFC 2326.

El reproductor de medios tiene cuatro trabajos principales:

1. Administrar la interfaz de usuario.
2. Manejar los errores de transmisión.
3. Descomprimir la música.
4. Eliminar la fluctuación.

En la actualidad, la mayoría de los reproductores de medios tienen una interfaz de usuario brillante que algunas veces simula una unidad de estéreo, con botones, palancas, barras deslizantes y despliegues visuales. Por lo general hay paneles frontales intercambiables, llamados máscaras (*skins*), que el usuario puede colocar en el reproductor. El reproductor de medios tiene que manejar todo esto e interactuar con el usuario.

Su segundo trabajo es tratar con los errores. La transmisión de música en tiempo real raramente utiliza TCP porque un error y una retransmisión podrían introducir un hueco demasiado grande en la música. En su lugar, la transmisión real por lo común se realiza con un protocolo como RTP,

el cual estudiamos en el capítulo 6. Al igual que la mayoría de los protocolos en tiempo real, la capa de RTP se encuentra encima de UDP, por lo que los paquetes pueden perderse. El reproductor es quien tiene que tratar esto.

En algunos casos, la música se intercala para facilitar el manejo de errores. Por ejemplo, un paquete podría contener 220 muestras de estéreo, cada una con un par de números de 16 bits, lo que normalmente está bien para 5 mseg de música. Pero tal vez el protocolo envíe todas las muestras impares en un intervalo de 10 mseg en un paquete, y todas las muestras pares en el siguiente. Por lo tanto, un paquete perdido no representa un hueco de 5 mseg en la música, sino la pérdida de cualquier otra muestra durante 10 mseg. Esta pérdida puede manejarse fácilmente haciendo que el reproductor de medios realice una interpolación mediante las muestras anterior y posterior para estimar el valor faltante.

En la figura 7-60 se ilustra el uso de intercalación para la recuperación de errores. Cada paquete contiene las muestras de tiempo alternadas durante un intervalo de 10 mseg. En consecuencia, perder el paquete 3, como se muestra, no crea un hueco en la música, sólo reduce la resolución temporal por algún tiempo. Los valores faltantes pueden interpolarse para proporcionar música continua. Este esquema particular sólo funciona con el muestreo sin comprimir, pero muestra la forma en que un código adecuado puede hacer que un paquete perdido signifique menos calidad en lugar de un hueco de tiempo. Sin embargo, el RFC 3119 proporciona un esquema que funciona con el audio comprimido.

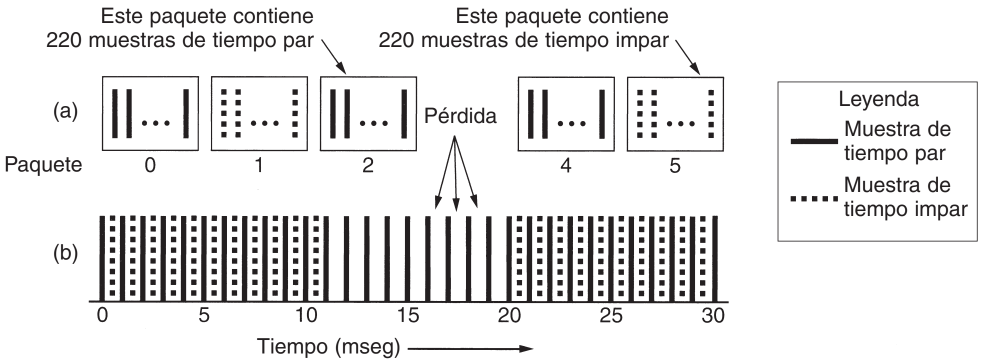


Figura 7-60. Cuando los paquetes transportan muestras alternas, la pérdida de un paquete reduce la resolución temporal en lugar de crear un hueco en el tiempo.

El tercer trabajo del reproductor de medios es descomprimir la música. Aunque esta tarea es intensa para la computadora, es muy directa.

El cuarto trabajo es eliminar la fluctuación, el veneno de todos los sistemas en tiempo real. Todos los sistemas de audio de flujo continuo inician almacenando en el búfer aproximadamente de 10 a 15 seg de música antes de comenzar a reproducir, como se muestra en la figura 7-61. Idealmente, el servidor continuará llenando el búfer a la tasa exacta a la que el reproductor de medios lo vacía, aunque en realidad esto no podría suceder, por lo que la retroalimentación en el ciclo podría ser útil.

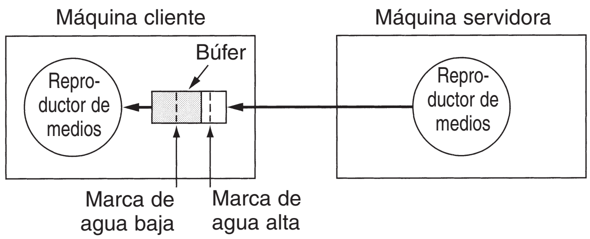


Figura 7-61. El reproductor de medios almacena en el búfer la entrada del servidor de medios y reproduce desde el servidor en lugar de hacerlo directamente desde la red.

Se pueden utilizar dos métodos para mantener lleno el búfer. Con un **servidor *pull*** (de recepción automática), siempre y cuando haya espacio en el búfer para otro bloque, el reproductor de medios simplemente sigue enviando al servidor mensajes en los que le solicita un bloque adicional. Su objetivo es mantener el búfer lo más lleno posible.

La desventaja de un servidor *pull* son todas las solicitudes de datos innecesarias. El servidor sabe que ha enviado el archivo completo, de modo que, ¿por qué el reproductor sigue enviando solicitudes? Por esta razón, raramente se utiliza.

Con un **servidor *push*** (de actualización automática), el reproductor de medios envía una solicitud *PLAY* y el servidor simplemente continúa enviándole datos. Aquí hay dos posibilidades: el servidor de medios se ejecuta a la velocidad normal de reproducción o se ejecuta más rápido. En ambos casos, algunos datos se almacenan en el búfer antes de que inicie la reproducción. Si el servidor se ejecuta a la velocidad normal de reproducción, los datos que provengan de él se agregan al final del búfer y el reproductor elimina los datos del frente del búfer para reproducirlos. Siempre y cuando todo funcione a la perfección, la cantidad de datos en el búfer permanece constante. Este esquema es sencillo debido a que no se necesitan mensajes de control en ninguna dirección.

El otro método *push* es hacer que el servidor envíe datos a una velocidad mayor que la necesaria. La ventaja aquí es que si no se puede garantizar que el servidor se ejecute a una tasa regular, tiene la oportunidad de reponerse si se queda atrás. Sin embargo, un problema aquí son los desbordamientos de búfer potenciales si el servidor puede enviar datos con más rapidez que con la que se consumen (y debe poder hacer esto para evitar los huecos).

La solución es que el reproductor de medios defina una **marca de agua baja** y una **marca de agua alta** en el búfer. Básicamente, el servidor sólo envía datos hasta que el búfer llega a la marca de agua alta. A continuación el reproductor de medios le indica que haga una pausa. Puesto que los datos continuarán llegando hasta que el servidor obtenga la solicitud de pausa, la distancia entre la marca de agua alta y el final del búfer tiene que ser mayor que el producto del retardo de ancho de banda de la red. Después de que el servidor se detenga, el búfer comenzará a vaciarse. Cuando llegue a la marca de agua baja, el reproductor de medios indicará al servidor de medios que comience de nuevo. La marca de agua baja tiene que colocarse de manera que la subutilización de búfer no ocurra.

Para operar un servidor *push*, el reproductor de medios necesita un control remoto para él. RTSP, que se define en el RFC 2326, proporciona el mecanismo para que el reproductor controle al servidor. No proporciona el flujo de datos, que por lo general es RTP. En la figura 7-62 se listan los principales comandos que proporciona RTSP.

Comando	Acción del servidor
DESCRIBE	Lista los parámetros de los medios
SETUP	Establece un canal lógico entre el reproductor y el servidor
PLAY	Comienza a enviar los datos al cliente
RECORD	Comienza a aceptar los datos del cliente
PAUSE	Detiene de manera temporal el envío de datos
TEARDOWN	Libera el canal lógico

Figura 7-62. Los comandos RTSP del reproductor al servidor.

7.4.4 Radio en Internet

Una vez que pudo ser posible difundir audio a través de Internet, las estaciones de radio comerciales tuvieron la idea de transmitir su contenido a través de Internet, así como a través de aire. No mucho tiempo después, las estaciones de radio universitarias comenzaron a colocar su señal en Internet. Después, los *estudiantes* comenzaron sus propias estaciones de radio. Con la tecnología actual, casi cualquier persona puede iniciar una estación de radio. El área de la radio de Internet es muy nueva y se encuentra en un proceso de cambio, pero vale la pena decir un poco más.

Hay dos métodos generales para la radio en Internet. En el primero, los programas se graban con anterioridad y se almacenan en disco. Los escuchas pueden conectarse a los archivos de la estación de radio y obtener y descargar cualquier programa para escucharlo. De hecho, esto es exactamente lo mismo que el audio de flujo continuo que acabamos de analizar. También es posible almacenar cada programa justo después de que se transmite en vivo, por lo que el archivo sólo está atrasado, digamos, media hora, o menos con respecto de la transmisión en vivo. Este método tiene las ventajas de que es fácil de llevar a cabo, de que las demás técnicas que hemos visto hasta aquí también lo son y de que los escuchas pueden elegir de entre todos los programas del archivo.

El otro método es difundir el contenido en vivo a través de Internet. Algunas estaciones transmiten a través de aire y a través de Internet de manera simultánea, pero cada vez hay más estaciones de radio que sólo transmiten a través de Internet. Algunas de las técnicas que se aplican al audio de flujo continuo también se aplican a la radio en vivo por Internet, pero también hay algunas diferencias clave.

Un punto que es el mismo es la necesidad de almacenar en el búfer del usuario para disminuir la fluctuación. Al colectar 10 o 15 segundos de radio antes de comenzar la reproducción, el audio puede escucharse bien, aunque suceda fluctuación sustancial a través de la red. En tanto todos los paquetes lleguen antes de que se necesiten, no importa cuándo lleguen.

Una diferencia clave es que el audio de flujo continuo puede enviarse a una tasa mayor que la de reproducción, puesto que el receptor puede detenerla cuando se llegue a la marca de agua alta. Potencialmente, esto le da el tiempo para retransmitir los paquetes perdidos, aunque esta estrategia no es muy común. En contraste, la radio en vivo siempre se difunde a la tasa exacta a la que se genera y a la que se reproduce.

Otra diferencia es que una estación de radio por lo general tiene cientos o miles de escuchas simultáneos mientras que el audio de flujo continuo es de punto a punto. Bajo estas circunstancias, la radio en Internet debería utilizar multidifusión con los protocolos RTP/RTSP. Ésta es claramente la forma más eficiente de operar.

En la actualidad, la radio en Internet no funciona así. Lo que sucede realmente es que el usuario establece una conexión TCP con la estación y la alimentación se envía a través de una conexión TCP. Por supuesto, esto crea varios problemas, como que el flujo se detenga cuando la ventana esté llena, que los paquetes perdidos expiren y se retransmitan, etcétera.

Hay tres razones por las que se utiliza la unidifusión TCP en lugar de la multidifusión RTP. La primera es que pocos ISPs soportan la multidifusión, por lo que no es una opción práctica. La segunda es que RTP es menos conocido que TCP y las estaciones de radio por lo general son pequeñas y tienen poca experiencia en computación, por lo que es más fácil que utilicen un protocolo que conocen bien y que es soportado por todos los paquetes de software. La tercera es que muchas personas escuchan la radio en Internet en su trabajo, lo que en la práctica significa detrás de un *firewall*. La mayoría de los administradores de sistemas configura su *firewall* para proteger su LAN contra visitantes no bienvenidos. Por lo general, tales administradores permiten conexiones TCP del puerto remoto 25 (SMTP para correo electrónico), paquetes UDP del puerto remoto 53 (DNS) y conexiones TCP del puerto remoto 80 (HTTP para Web). Casi todo lo demás podría bloquearse, incluyendo RTP. Por lo tanto, la única forma de obtener la señal de radio a través del *firewall* es que el sitio Web pretenda ser un servidor HTTP, por lo menos ante el *firewall*, y que utilice servidores HTTP, los cuales hablan TCP. Aunque estas medidas severas proporcionan un mínimo de seguridad, por lo general obligan a las aplicaciones multimedia a que utilicen modos de operación drásticamente menos eficientes.

Puesto que la radio en Internet es un medio nuevo, las guerras de los formatos están en su apogeo. RealAudio, Windows Media Audio y MP3 están compitiendo de manera agresiva en este mercado para ser el formato dominante para la radio en Internet. Un recién llegado es Vorbis, que técnicamente es similar a MP3 pero es código abierto y tiene las diferencias suficientes como para no utilizar las patentes en la que se basa MP3.

Una estación de radio típica de Internet tiene una página Web que lista su agenda, información sobre sus DJs y anunciadores, y muchos anuncios. También hay varios iconos que listan los formatos de audio que la estación soporta (o simplemente ESCUCHAR AHORA si sólo soporta un formato). Estos iconos o ESCUCHAR AHORA están vinculados con metaarchivos del tipo que analizamos anteriormente.

Cuando un usuario hace clic en uno de estos iconos, se envía el pequeño metaarchivo. El navegador utiliza su tipo MIME o extensión de archivo para determinar la aplicación auxiliar apropiada (es decir, el reproductor de medios) para el metaarchivo. A continuación escribe el metaarchivo en un archivo de trabajo en disco, inicia el reproductor de medios y le entrega el nombre del archivo de trabajo. El reproductor de medios lee dicho archivo, ve el URL que contiene (por lo general, con un esquema *http* en lugar de *rtsp* para solucionar el problema del *firewall* y porque algunas aplicaciones populares de multimedia funcionan de esa manera), contacta al servidor y comienza a actuar como un radio. Además, el audio sólo tiene un flujo, por lo que *http* funciona, pero para el vídeo, que por lo menos tiene dos flujos, *http* no funciona y lo que realmente se necesita es algo como *rtsp*.

Otro desarrollo interesante en el área de la radio en Internet es un arreglo en el que cualquiera, incluso un estudiante, puede establecer y operar una estación de radio. Los componentes principales se ilustran en la figura 7-63. La base de la estación es una PC ordinaria con una tarjeta de sonido y un micrófono. El software consiste en un reproductor de medios, como Winamp o Freeamp, con un *plug-in* para la captura de audio y un codec para el formato de salida seleccionado, por ejemplo, MP3 o Vorbis.

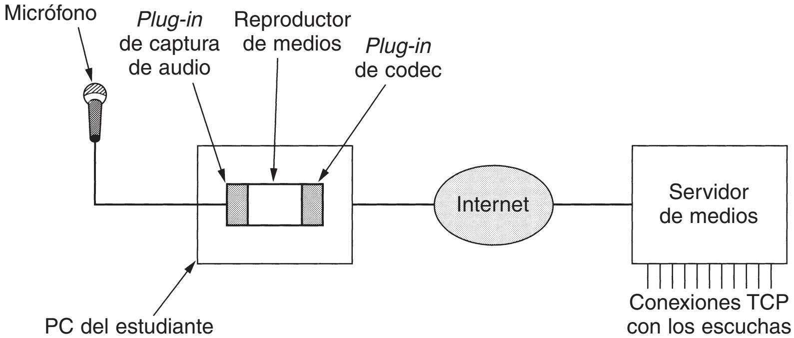


Figura 7-63. Una estación de radio estudiantil.

A continuación el flujo de audio generado por la estación se alimenta a través de Internet hacia un servidor grande, el cual lo distribuye a una gran cantidad de conexiones TCP. Por lo general, el servidor soporta muchas estaciones pequeñas. También mantiene un directorio de las estaciones que tiene y de lo que está actualmente en el aire en cada una. Los escuchas potenciales van al servidor, seleccionan una estación y obtienen una alimentación TCP. Hay paquetes de software comercial para manejar todas las piezas y paquetes de código abierto, como *icecast*. También hay servidores que están dispuestos a manejar la distribución a cambio de una cuota.

7.4.5 Voz sobre IP

En sus inicios, el sistema telefónico conmutado público se utilizaba principalmente para el tráfico de voz con un poco de tráfico de datos por aquí y por allá. Pero el tráfico de datos creció y creció, y aproximadamente en 1999, la cantidad de bits de datos movidos igualó a la de bits de voz (debido a que la voz está en PCM en las troncales, puede medirse en bits/seg). En el 2002, el volumen del tráfico de datos era mayor que el volumen del tráfico de voz y continúa creciendo de manera exponencial, mientras que el crecimiento del tráfico de voz casi era plano (con un crecimiento anual de 5%).

Como consecuencia de estas cifras, muchos operadores de redes de conmutación de paquetes de repente se interesaron en transportar voz a través de sus redes de datos. La cantidad de ancho de banda adicional requerida para voz es minúscula debido a que las redes de paquetes están dimensionadas para el tráfico de datos. Sin embargo, probablemente el recibo telefónico de la persona promedio sea más grande que su cuenta de Internet, por lo que los operadores de redes de datos

vieron la telefonía de Internet como una forma de ganar una gran cantidad de dinero adicional sin tener que colocar una nueva fibra en el piso. De esta forma nació la **telefonía de Internet** (también conocida como **voz sobre IP).**

H.323

Lo que a todos les quedó claro desde el principio fue que si cada fabricante diseñaba su propia pila de protocolos, tal vez el sistema nunca funcionaría. Para evitar este problema, un grupo de personas interesadas se unieron bajo la protección de la ITU para trabajar en estándares. En 1996 la ITU emitió la recomendación **H.323** titulada "Sistemas Telefónicos Visuales y Equipo para Redes de Área Local que Proporcionan una Calidad de Servicio No Garantizada". Sólo la industria telefónica podría pensar en tal nombre. La recomendación se revisó en 1998, y esta H.323 revisada fue la base de los primeros sistemas de telefonía de Internet ampliamente difundidos.

H.323 es más una revisión arquitectónica de la telefonía de Internet que un protocolo específico. Hace referencia a una gran cantidad de protocolos específicos para codificación de voz, establecimiento de llamadas, señalización, transporte de datos y otras áreas, en lugar de especificar estas cosas en sí. El modelo general se ilustra en la figura 7-64. En el centro se encuentra una **puerta de enlace** que conecta Internet con la red telefónica. Dicha puerta de enlace habla los protocolos H.323 en el lado de Internet y los protocolos PSTN en el lado del teléfono. Los dispositivos de comunicación se llaman **terminales.** Una LAN podría tener un **gatekeeper,** el cual controla los puntos finales bajo su jurisdicción, llamados **zona.**

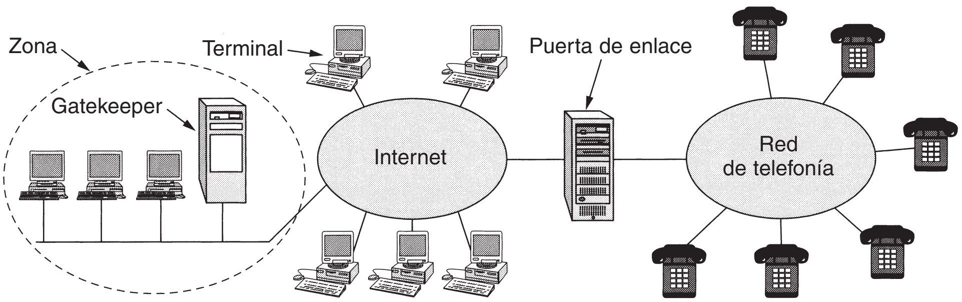


Figura 7-64. El modelo arquitectónico H.323 de la telefonía de Internet.

Una red telefónica necesita una cantidad de protocolos. Para empezar, hay un protocolo para codificar y decodificar voz. El sistema PCM que estudiamos en el capítulo 2 está definido en la recomendación **G.711** de la ITU. Codifica un solo canal de voz muestreando 8000 veces por segundo con una muestra de 8 bits para proporcionar voz descomprimida a 64 kbps. Todos los sistemas H.323 deben soportar G.711. Sin embargo, también se permiten otros protocolos de compresión de voz (aunque no son requeridos). Utilizan diferentes algoritmos de compresión y realizan diferentes intercambios entre calidad y ancho de banda. Por ejemplo, **G.723.1** toma un

bloque de 240 muestras (30 mseg de voz) y utiliza la codificación predictiva para reducirlo ya sea a 24 o a 20 bytes. Este algoritmo proporciona una tasa de salida ya sea de 6.4 o 5.3 kbps (factores de compresión de 10 y 12), respectivamente, con poca pérdida en la calidad percibida. También se permiten otros codecs.

Puesto que están permitidos múltiples algoritmos de compresión, se necesita un protocolo para permitir que las terminales negocien cuál van a utilizar. Este protocolo se llama **H.245.** También negocia otros aspectos de la conexión, como la tasa de bits. RTCP es necesario para el control de los canales RTP. También se necesita un protocolo para establecer y liberar conexiones, proporcionar tonos de marcado, emitir sonidos de marcado y el resto de la telefonía estándar. Aquí se utiliza **Q.931** de la ITU. Las terminales necesitan un protocolo para hablar con el *gatekeeper* (si está presente). Para este propósito se utiliza **H.225.** El canal PC a *gatekeeper* manejado por este protocolo se conoce como canal **RAS** (**Registro/Admisión/Estado**)**.** Este canal permite terminales para unirse y dejar la zona, solicitar y regresar ancho de banda, y proporcionar actualizaciones de estado, entre otras cosas. Por último, se necesita un protocolo para la transmisión real de datos. RTP se utiliza para este propósito. Es manejado por RTCP, como es usual. La posición de todos estos protocolos se muestra en la figura 7-65.

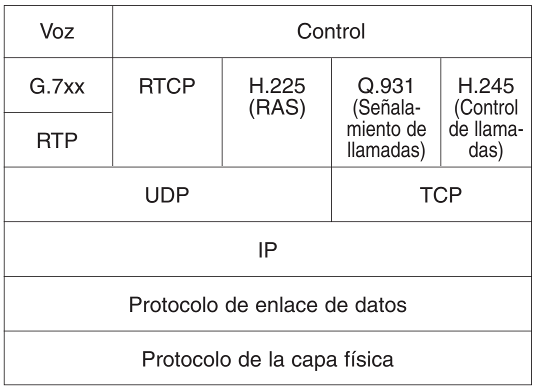


Figura 7-65. La pila de protocolos H.323.

Para ver cómo se ajustan estos protocolos, considere el caso de una terminal de PC en una LAN (con un *gatekeeper*) que llama a un teléfono remoto. La PC primero tiene que descubrir al *gatekeeper*, por lo que difunde un paquete UDP de descubrimiento de *gatekeeper* al puerto 1718. Cuando el *gatekeeper* responde, la PC se aprende la dirección IP de éste. Ahora la PC se registra con el *gatekeeper* enviándole un mensaje RAS en un paquete UDP. Después de que es aceptada, la PC envía al *gatekeeper* un mensaje de admisión RAS solicitando ancho de banda. Sólo después de que se ha proporcionado el ancho de banda, se puede establecer la llamada. La idea de solicitar ancho de banda con anticipación es permitir que el *gatekeeper* limite el número de llamadas para evitar sobrescribir la línea saliente para ayudar a proporcionar la calidad de servicio necesaria.

La PC ahora establece una conexión TCP con el *gatekeeper* para comenzar el establecimiento de la llamada. Este establecimiento utiliza los protocolos de red telefónicos existentes, que están orientados a la conexión, por lo que se necesita TCP. En contraste, el sistema telefónico no tiene nada parecido a RAS para permitir que los teléfonos anuncien su presencia, por lo que los diseñadores de H.323 eran libres de utilizar UDP o TCP para RAS, y eligieron el UDP que genera menor información adicional.

Ahora que tiene asignado ancho de banda, la PC puede enviar un mensaje *SETUP* de Q.931 a través de la conexión TCP. Este mensaje especifica el número del teléfono al que se está llamando (o la dirección IP y el puerto, si se está llamando a una computadora). El *gatekeeper* responde con un mensaje *CALL PROCEEDING* de Q.931 para confirmar la recepción de la solicitud. Enseguida el *gatekeeper* reenvía un mensaje *SETUP* a la puerta de enlace.

A continuación, la puerta de enlace, que es mitad computadora y mitad conmutador de teléfono, realiza una llamada telefónica ordinaria al teléfono deseado (ordinario). La oficina central a la que está anexado el teléfono hace que suene el teléfono al que se hace la llamada y también regresa un mensaje *ALERT* de Q.931 para indicar a la PC invocadora que ya se ha emitido el sonido. Cuando la persona en el otro extremo levanta el teléfono, la oficina central regresa un mensaje *CONNECT* de Q.931 para indicar a la PC que tiene una conexión.

Una vez que se ha establecido la conexión, el *gatekeeper* ya no está en el ciclo, aunque la puerta de enlace sí lo está, por supuesto. Los paquetes subsiguientes ignoran al *gatekeeper* y van directo a la dirección IP de la puerta de enlace. En este punto, simplemente tenemos un tubo que se ejecuta entre los dos lados. Ésta es tan sólo una conexión de capa física para mover bits, nada más. Ninguno de los lados sabe nada del otro.

A continuación se utiliza el protocolo H.245 para negociar los parámetros de la llamada. Utiliza el canal de control H.245, que siempre está abierto. Cada lado inicia anunciando sus capacidades, por ejemplo, si puede manejar vídeo (H.323 puede manejar vídeo) o llamadas de conferencia, qué codecs soporta, etcétera. Una vez que cada lado sabe lo que el otro puede manejar, se establecen dos canales de datos unidireccionales y a cada uno se le asigna un codec y otros parámetros. Puesto que cada lado puede tener equipo diferente, es posible que los codecs en los canales hacia adelante e inverso sean diferentes. Una vez que se terminan todas las negociaciones, puede comenzar el flujo de datos utilizando RTP. Tal flujo se maneja mediante RTCP, que juega un papel en el control de congestionamiento. Si el vídeo está presente, RTCP maneja la sincronización de audio/vídeo. En la figura 7-66 se muestran los diversos canales. Cuando cualquiera de las dos partes cuelga, se utiliza el canal de señalización de llamada de Q.931 para terminar la conexión.

Cuando se termina la llamada, la PC invocadora contacta al *gatekeeper* nuevamente con un mensaje RAS para liberar el ancho de banda que se ha asignado. De manera alterna, puede realizar otra llamada.

No hemos mencionado nada sobre la calidad del servicio, aunque ésta es esencial para que la voz sobre IP sea un éxito. La razón es que la QoS está más allá del alcance de H.323. Si la red subyacente es capaz de producir una conexión estable, libre de fluctuación de la PC invocadora (por ejemplo, utilizando las técnicas que analizamos en el capítulo 5) a la puerta de enlace, entonces la QoS de la llamada será buena; de lo contrario, no lo será. La parte del teléfono utiliza PCM y siempre está libre de fluctuación.

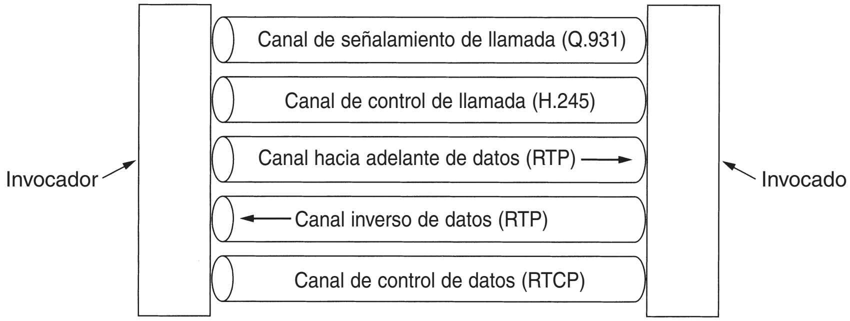


Figura 7-66. Canales lógicos entre el invocador y el invocado durante una llamada.

SIP—Protocolo de Inicio de Sesión

H.323 fue diseñado por la ITU. Muchas personas de la comunidad de Internet lo vieron como un producto típico telco: grande, complejo e inflexible. En consecuencia, la IETF estableció un comité para diseñar una forma más simple y modular para transmitir voz sobre IP. El resultado principal hasta la fecha es el **SIP** (**Protocolo de Inicio de Sesión**)**,** que se describe en el RFC 3261. Este protocolo describe cómo establecer llamadas telefónicas a Internet, videoconferencias y otras conexiones multimedia. A diferencia de H.323, que es un conjunto completo de protocolos, SIP está compuesto de un solo módulo, pero se ha diseñado para que interactúe bien con las aplicaciones de Internet existentes. Por ejemplo, define los números telefónicos como URLs, a fin de que las páginas Web puedan contenerlos, lo que permite hacer clic en un vínculo para iniciar una llamada telefónica (de la misma forma en que un esquema *mailto* permite hacer clic en un vínculo que despliega un programa para enviar un mensaje de correo electrónico).

SIP puede establecer sesiones de dos partes (llamadas de teléfono ordinarias), de múltiples partes (en donde todos pueden oír y hablar) y de multidifusión (un emisor, muchos receptores). Las sesiones pueden contener audio, vídeo o datos; las de multidifusión son útiles para juegos de múltiples jugadores, por ejemplo. SIP sólo maneja establecimiento, manejo y terminación de sesiones. Para el transporte de datos, se utilizan otros protocolos, como RTP/RTCP. SIP es un protocolo de capa de aplicación y puede ejecutarse sobre UDP o TCP.

SIP soporta una variedad de servicios, como localizar al invocado (quien tal vez no esté en su máquina doméstica) y determinar las capacidades de éste, así como manejar los mecanismos del establecimiento y la terminación de la llamada. En el caso más simple, SIP establece una sesión de la computadora del invocador a la del invocado, por lo que primero analizaremos ese caso.

Los números telefónicos de SIP se representan como URLs que utilizan el esquema *sip*, por ejemplo, *sip:ilse@cs.university.edu* para un usuario de nombre Ilse en el *host* especificado por el nombre DNS *cs.university.edu*. Los URLs de SIP también pueden contener direcciones IPv4, IPv6 o números telefónicos reales.

El protocolo SIP se basa en texto y está modelado en HTTP. Una parte envía un mensaje en texto ASCII que consiste en un nombre de método en la primera línea, seguido por líneas adicionales que contienen encabezados para pasar los parámetros. Muchos de estos encabezados se toman de MIME para permitir que SIP interactúe con las aplicaciones de Internet existentes. En la figura 7-67 se listan los seis métodos definidos por la especificación central.

Método	Descripción
INVITE	Solicita el inicio de una sesión
ACK	Confirma que se ha iniciado una sesión
BYE	Solicita la terminación de una sesión
OPTIONS	Consulta a un *host* sobre sus capacidades
CANCEL	Cancela una solicitud pendiente
REGISTER	Informa a un servidor de redireccionamiento sobre la ubicación actual del usuario

Figura 7-67. Los métodos SIP definidos en la especificación central.

Para establecer una sesión, el invocador crea una conexión TCP con el invocado y envía un mensaje *INVITE* a través de ella o lo envía en un paquete UDP. En ambos casos, los encabezados de la segunda línea y de las subsiguientes describen la estructura del cuerpo del mensaje, el cual contiene las capacidades, los tipos de medios y los formatos del invocador. Si el invocado acepta la llamada, responde con un código de respuesta tipo HTTP (un número de tres dígitos que utiliza los grupos de la figura 7-42, 200 para aceptación). El invocado también puede proporcionar información sobre sus capacidades, tipos de medios y formatos, después de la línea de código de respuesta.

La conexión se realiza utilizando un acuerdo de tres vías, de modo que el invocador responde con un mensaje *ACK* para terminar el protocolo y confirmar la recepción del mensaje 200.

Cualquiera de las dos partes puede solicitar la terminación de una sesión enviando un mensaje que contiene el método *BYE*. Cuando el otro lado confirma su recepción, se termina la sesión.

El método *OPTIONS* se utiliza para consultar a una máquina sobre sus propias capacidades. Por lo general, se utiliza antes de que se inicie una sesión a fin de averiguar si esa máquina tiene la capacidad de transmitir voz sobre IP o el tipo de sesión que se ha contemplado.

El método *REGISTER* se relaciona con la capacidad de SIP de rastrear y conectarse con un usuario que esté lejos de casa. Este mensaje se envía a un servidor de ubicación SIP que mantiene la pista de quién está en qué lugar. Ese servidor se puede consultar posteriormente para encontrar la ubicación actual del usuario. En la figura 7-68 se ilustra la operación de redirección. Aquí el invocador envía el mensaje *INVITE* a un servidor *proxy* para ocultar la posible redirección. A continuación el *proxy* investiga en dónde está el usuario y envía el mensaje *INVITE* ahí. Después actúa como un regulador para los mensajes subsecuentes en el acuerdo de tres vías. Los mensajes *LOOKUP* y *REPLY* no son parte de SIP; se puede utilizar cualquier protocolo conveniente, dependiendo del tipo de servidor de ubicación que se esté utilizando.

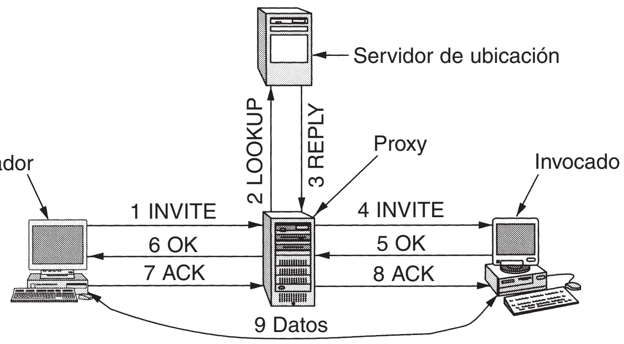


Figura 7-68. Uso de servidores proxy y de redirección con SIP.

SIP tiene una variedad de otras características que no describiremos aquí, entre ellas la espera de llamadas, bloque de llamada, codificación y autenticación. También tiene la capacidad de colocar llamadas de una computadora en un teléfono ordinario, si está disponible una puerta de enlace entre Internet y el sistema telefónico.

Comparación entre H.323 y SIP

H.323 y SIP tienen muchas similitudes pero también algunas diferencias. Ambos permiten llamadas de dos partes y múltiples partes utilizando las computadoras y los teléfonos como puntos finales. Ambos soportan negociación de parámetros, codificación y los protocolos RTP/RTCP. En la figura 7-69 se muestra un resumen de las similitudes y diferencias.

Aunque los conjuntos de características son similares, los dos protocolos difieren ampliamente en la filosofía. H.323 es un estándar típico de peso completo de la industria de la telefonía, que especifica toda la pila de protocolos y que define de manera precisa lo que está permitido y lo que está prohibido. Este enfoque produce protocolos bien definidos en cada capa, lo que facilita la tarea de interoperabilidad. El precio pagado es un estándar grande, complejo y rígido que es difícil de adaptar a aplicaciones futuras.

En contraste, SIP es un protocolo de Internet típico que funciona intercambiando líneas cortas de texto ASCII. Es un módulo de carga ligera que interactúa bien con otros protocolos de Internet pero no tan bien con los de señalización de sistemas telefónicos. Debido a que el modelo IETF de voz sobre IP es altamente modular, es flexible y se puede adaptar con facilidad a las nuevas aplicaciones. La desventaja son problemas potenciales de interoperabilidad, aunque éstos se solucionan realizando reuniones frecuentes en las que los diferentes implementadores se reúnen para probar sus sistemas.

La voz sobre IP es un tema prometedor. En consecuencia, hay varios libros sobre él. Algunos ejemplos son (Collins, 2001; Davidson y Peters, 2000; Kumar y cols., 2001, y Wright, 2001). La edición de mayo/junio de 2002 de *Internet Computing* tiene varios artículos sobre este tema.

Elemento	H.323	SIP
Diseñado por	ITU	IETF
Compatibilidad con PSTN	Sí	Ampliamente
Compatibilidad con Internet	No	Sí
Arquitectura	Monolítica	Modular
Integridad	Pila de protocolos completa	SIP sólo maneja el establecimiento
Negociación de parámetros	Sí	Sí
Señalamiento de llamadas	Q.931 sobre TCP	SIP sobre TCP o UDP
Formato de mensajes	Binario	ASCII
Transporte de medios	RTP/RTCP	RTP/RTCP
Llamadas de múltiples partes	Sí	Sí
Conferencias multimedia	Sí	No
Direccionamiento	*Host* o número telefónico	URL
Terminación de llamadas	Explícita o liberación de TCP	Explícita o terminación de temporizador
Mensajes instantáneos	No	Sí
Encriptación	Sí	Sí
Tamaño de los estándares	1400 páginas	250 páginas
Implementación	Grande y compleja	Moderada
Estado	Distribuido ampliamente	Prometedor

Figura 7-69. Comparación entre H.323 y SIP.

7.4.6 Introducción al vídeo

Ya analizamos el oído hasta ahora; es tiempo de que analicemos el ojo (no, a esta sección no le sigue una sobre la nariz). El ojo humano tiene la propiedad de que, cuando una imagen incide en la retina, se retiene durante algunos milisegundos antes de desaparecer. Si una secuencia de imágenes incide a 50 o más imágenes/seg, el ojo no nota que está viendo imágenes discretas. Todos los sistemas de vídeo (es decir, televisión) aprovechan este principio para producir imágenes en movimiento.

Sistemas analógicos

Para entender los sistemas de vídeo, es mejor comenzar por la anticuada y sencilla televisión en blanco y negro. Para representar la imagen bidimensional que está frente a ella como un voltaje unidimensional en función del tiempo, la cámara barre rápidamente un haz de electrones a lo ancho de la imagen y lentamente hacia abajo, registrando la intensidad de la luz a su paso. Al final del barrido, llamado **trama,** el haz hace un retrazado. Esta intensidad como función de tiempo se difunde, y los receptores repiten el proceso de barrido para reconstruir la imagen. En la figura 7-70 se muestra el patrón de barrido que usan tanto la cámara como el receptor. (Como nota,

las cámaras CCD integran en lugar de barrer, pero algunas cámaras y todos los monitores hacen un barrido.)

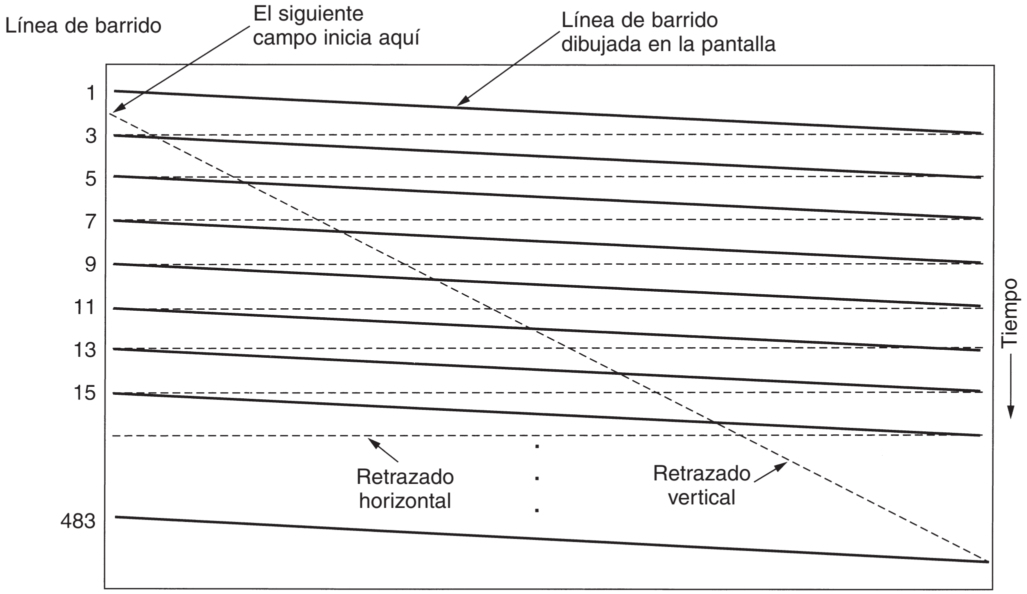


Figura 7-70. Patrón de barrido usado para el vídeo y la televisión NTSC.

Los parámetros de barrido exactos varían de país en país. El sistema usado en Norte y Sudamérica y en Japón tiene 525 líneas de barrido, una relación de aspecto horizontal a vertical de 4:3 y 30 tramas/seg. El sistema europeo tiene 625 líneas de barrido, la misma relación 4:3 y 25 tramas/seg. En ambos sistemas, no se presentan unas cuantas líneas de arriba y abajo (para aproximar una imagen rectangular a los CRTs originales, que eran redondos). Sólo se presentan 483 de las 525 líneas NTSC (y 576 de las 625 líneas de barrido PAL/SECAM). El haz se apaga durante el retrazado vertical, por lo que muchas estaciones (especialmente en Europa) usan este intervalo para difundir TeleTexto (páginas de texto que contienen noticias, informes meteorológicos, deportes, precios de acciones, etcétera).

Aunque 25 tramas/seg es suficiente para capturar una imagen continua, con esa tasa de tramas mucha gente, especialmente las personas mayores, percibe un parpadeo en la imagen (porque las imágenes viejas se desvanecen de la retina antes de la aparición de las nuevas). En lugar de aumentar la tasa de tramas, lo que requeriría usar más del escaso ancho de banda, se emplea un enfoque diferente. En lugar de presentar las líneas de barrido en orden, primero se despliegan las líneas de barrido nones, y luego las líneas de barrido pares. Cada una de estas medias tramas se llama **campo.** Hay experimentos que muestran que, aunque la gente nota el parpadeo a 25 tramas/seg, no lo nota a 50 campos/seg. Esta técnica se llama **entrelazado.** Se dice que la televisión (o vídeo) no entrelazada es **progresiva.** Observe que las películas se ejecutan a 24 fps, pero cada trama es completamente visible durante 1/24 seg.

El vídeo de color usa el mismo patrón de barrido que el monocromático (blanco y negro), excepto que, en lugar de desplegar la imagen mediante un solo haz en movimiento, se usan tres haces que se mueven al unísono. Se usa un haz para cada uno de los colores primarios aditivos: rojo, verde y azul (RGB). Esta técnica funciona porque puede construirse cualquier color a partir de la superposición lineal de rojo, verde y azul con las intensidades apropiadas. Sin embargo, para la transmisión por un solo canal, las tres señales de color deben combinarse en una sola señal **compuesta.**

Cuando se inventó la televisión a color, eran técnicamente posibles varios métodos para desplegar colores, así que varios países tomaron decisiones diferentes, lo que condujo a sistemas que aún son incompatibles. (Es importante mencionar que estas decisiones no tienen nada que ver con VHS contra Betamax contra P2000, que son métodos de grabación.) En todos los países, un requisito político fue que todos los programas que se transmitieran a color tenían que poder recibirse en los televisores existentes de blanco y negro. En consecuencia, no era aceptable el esquema más sencillo, codificar por separado las señales RGB. Sin embargo, RGB tampoco es el esquema más eficiente.

El primer sistema de color fue estandarizado en Estados Unidos por el **Comité Nacional de Estándares de Televisión**, que presentó sus siglas al estándar: **NTSC.** La televisión a color se introdujo en Europa varios años después, y para entonces la tecnología había mejorado de manera significativa, conduciendo a sistemas con mejor inmunidad contra el ruido y mejores colores. Éstos se llaman **SECAM** (**color secuencial con memoria**)**,** que se usa en Francia y Europa Oriental, y **PAL** (**línea de fases alternas**)**,** usado en el resto de Europa. La diferencia en la calidad del color entre NTSC y PAL/SECAM ha producido una broma de la industria que sugiere que NTSC significa en realidad Nunca Tendrás Semejanza en los Colores.

Para que las transmisiones puedan verse en receptores de blanco y negro, los tres sistemas combinan linealmente las señales RGB en una señal de **luminancia** (brillo) y dos de **crominancia** (color), aunque usan diferentes coeficientes para construir estas señales a partir de las señales RGB. Resulta interesante que el ojo es mucho más sensible a la señal de luminancia que a las de crominancia, por lo que estas últimas no necesitan transmitirse con tanta precisión. En consecuencia, la señal de luminancia puede difundirse a la misma frecuencia que la vieja señal de blanco y negro, y puede recibirse en los televisores de blanco y negro. Las dos señales de crominancia se difunden en bandas angostas a frecuencias mayores. Algunos aparatos de televisión tienen controles etiquetados brillo, matiz y saturación (o brillo, tinte y color) para controlar estas tres señales por separado. Es necesario el entendimiento de la luminancia y la crominancia para comprender el funcionamiento de la compresión de vídeo.

En los últimos años ha habido un interés considerable en la **HDTV** (**televisión de alta definición**)**,** que produce imágenes más nítidas al duplicar (aproximadamente) la cantidad de líneas de barrido. Estados Unidos, Europa y Japón han desarrollado sistemas HDTV, todos diferentes y todos mutuamente incompatibles. ¿Usted esperaba otra cosa? Los principios básicos de la HDTV en términos de barrido, luminancia, crominancia, etcétera, son semejantes a los sistemas actuales. Sin embargo, los tres formatos tienen una relación de aspecto común de 16:9 en lugar 4:3 para lograr una correspondencia mejor con el formato usado para el cine (que se graba en película de 35 mm, y tiene una relación de aspecto de 3:2).

Sistemas digitales

La representación más sencilla del vídeo digital es una secuencia de tramas, cada una de las cuales consiste en una malla rectangular de elementos de imagen, o **píxeles.** Cada píxel puede ser un solo bit, para representar blanco o negro. La calidad de tal sistema es parecida a la que se obtiene al enviar una fotografía a color por fax: espantosa. (Inténtelo si puede; de lo contrario, fotocopie una fotografía a color en una máquina copiadora que no maneje tonos.)

El siguiente paso es usar ocho bits por píxel para representar 256 niveles de gris. Este esquema da vídeo en blanco y negro de alta calidad. Para vídeo a color, los sistemas buenos usan 8 bits por cada uno de los colores RGB, aunque casi todos los sistemas los mezclan en vídeo compuesto para su transmisión. Aunque el uso de 24 bits por píxel limita la cantidad de colores a unos 16 millones, el ojo humano no puede diferenciar tantos colores. Las imágenes digitales de color se producen usando tres haces de barrido, uno por color. La geometría es la misma que en el sistema analógico de la figura 7-70, excepto que las líneas continuas de barrido ahora se reemplazan por elegantes filas de píxeles discretos.

Para producir una imagen uniforme, el vídeo digital, al igual que el vídeo analógico, debe presentar cuando menos 25 tramas/seg. Sin embargo, dado que los monitores de computadora de alta calidad con frecuencia barren la pantalla a partir de las imágenes almacenadas en memoria a razón de 75 veces por segundo o más, no se requiere entrelazado y, en consecuencia, normalmente no se usa. Basta con repintar (es decir, redibujar) la misma trama tres veces consecutivas para eliminar el parpadeo.

En otras palabras, la uniformidad de movimiento es determinada por la cantidad de imágenes *diferentes* por segundo, mientras que el parpadeo es determinado por la cantidad de veces por segundo que se pinta la trama. Estos dos parámetros son diferentes. Una imagen fija pintada a 20 tramas/seg no mostrará un movimiento inestable, pero parpadeará porque una trama se desvanecerá de la retina antes de que aparezca la siguiente. Una película con 20 tramas diferentes por segundo, cada una pintada cuatro veces seguidas, no parpadeará, pero el movimiento parecerá no uniforme.

El significado de estos dos parámetros se aclara cuando consideramos el ancho de banda necesario para transmitir vídeo digital a través de una red. Todos los monitores de computadora actuales usan la relación de aspecto 4:3 para poder usar tubos de rayos catódicos económicos de producción en masa diseñados para el mercado de televisión de consumidor. Las configuraciones comunes son 1024×768, 1280×960 y 1600×1200. Incluso la más pequeña de éstas con 24 bits por píxel y 25 tramas/seg necesita alimentarse a 472 Mbps. Una portadora OC-12 de SONET tendría que manejar esta situación, y la ejecución de este tipo de portadora en la casa de todas las personas aún no es posible. La duplicación de esta tasa para evitar el parpadeo es aún menos atractiva. Una mejor solución es transmitir 25 tramas/seg y hacer que la computadora almacene cada una y la pinte dos veces. La televisión difundida no usa esta estrategia porque las televisiones no tienen memoria y, aunque la tuvieran, para almacenarse en RAM, las señales analógicas primero se tendrían que convertir a un formato digital, lo que requeriría hardware extra. Como consecuencia, se necesita el entrelazado para la televisión difundida, pero no para el vídeo digital.

7.4.7 Compresión de vídeo

A estas alturas debe ser obvio que la transmisión de vídeo sin comprimir es impensable. La única esperanza es que sea posible la compresión masiva. Por fortuna, durante las últimas décadas un gran número de investigaciones ha conducido a muchas técnicas y algoritmos de compresión que hacen factible la transmisión de vídeo. En esta sección estudiaremos cómo se consigue la compresión de vídeo.

Todos los sistemas de compresión requieren dos algoritmos: uno para la compresión de los datos en el origen y otro para la descompresión en el destino. En la literatura, estos algoritmos se conocen como algoritmos de **codificación** y **decodificación,** respectivamente. También usaremos esta terminología aquí.

Estos algoritmos tienen ciertas asimetrías y es importante comprenderlas. Primero, en muchas aplicaciones un documento multimedia, digamos una película, sólo se codificará una vez (cuando se almacene en el servidor multimedia), pero se decodificará miles de veces (cuando los clientes lo vean). Esta asimetría significa que es aceptable que el algoritmo de codificación sea lento y requiera hardware costoso, siempre y cuando el algoritmo de decodificación sea rápido y no requiera hardware costoso. A fin de cuentas, el operador de un servidor multimedia podría esta dispuesto a rentar una supercomputadora paralela durante algunas semanas para codificar su biblioteca de vídeo completa, pero requerir que los consumidores renten una supercomputadora durante dos horas para ver un vídeo seguramente no tendrá mucho éxito. Muchos sistemas de compresión prácticos llegan a extremos considerables para lograr que la decodificación sea rápida y sencilla, aun al precio de hacer lenta y complicada la codificación.

Por otra parte, para la multimedia en tiempo real, como las videoconferencias, la codificación lenta es inaceptable. La codificación debe ocurrir al momento, en tiempo real. En consecuencia, la multimedia en tiempo real usa algoritmos o parámetros diferentes que el almacenamiento de vídeos en disco, lo que con frecuencia resulta en una compresión significativamente menor.

Una segunda asimetría es que no es necesaria la capacidad de invertir el proceso de codificación/decodificación. Es decir, al comprimir, transmitir y descomprimir un archivo de datos, el usuario espera recibir el original, correcto hasta el último bit. En multimedia este requisito no existe. Por lo general, es aceptable que la señal de vídeo después de codificar y decodificar sea ligeramente diferente de la original. Cuando la salida decodificada no es exactamente igual a la entrada original, se dice que el sistema es **con pérdida** (*lossy*). Si la entrada y la salida son idénticas, el sistema es **sin pérdida** (*lossless*). Los sistemas con pérdida son importantes porque aceptar una pequeña pérdida de información puede ofrecer ventajas enormes en términos de la posible relación de compresión.

Estándar JPEG

Un vídeo es sólo una secuencia de imágenes (más sonido). Si pudiéramos encontrar un buen algoritmo para codificar una sola imagen, tal algoritmo podría aplicarse a cada imagen en sucesión para alcanzar la compresión de vídeo. Existen algoritmos buenos de compresión de imágenes fijas, por lo que comenzaremos ahí nuestro estudio de la compresión de vídeo. El estándar **JPEG**

(**Grupo Unido de Expertos en Fotografía**) para la compresión de imágenes fijas de tono continuo (por ejemplo, fotografías) fue desarrollado por expertos en fotografía que trabajaban bajo la tutela de la ITU, la ISO y el IEC, otro grupo de estándares. Es importante para la multimedia porque, a primera vista, el estándar multimedia para imágenes en movimiento, MPEG, es simplemente la codificación JPEG por separado de cada trama, más otras características extra para la compresión entre tramas y la detección de movimiento. El JPEG se define en el Estándar Internacional 10918.

JPEG tiene cuatro modos y muchas opciones; se parece más a una lista de compras que a un algoritmo. No obstante, para nuestros fines sólo es relevante el modo secuencial con pérdida, y éste se ilustra en la figura 7-71. Además, nos concentraremos en la manera en que el JPEG se usa normalmente para codificar imágenes de vídeo RGB de 24 bits y dejaremos fuera algunos de los detalles menores por cuestiones de sencillez.

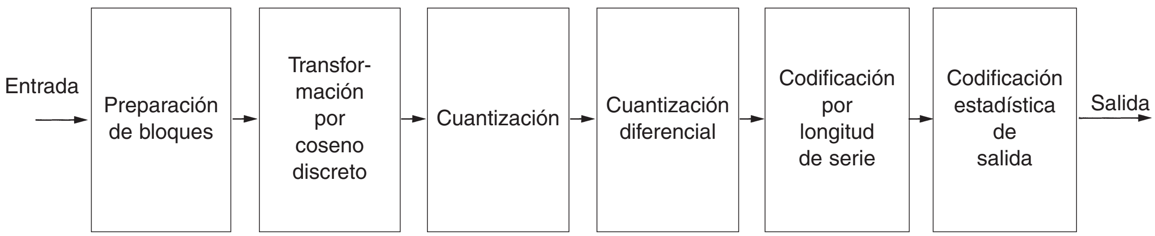


Figura 7-71. Operación de JPEG en el modo secuencial con pérdida.

El paso 1 de la codificación de una imagen con JPEG es la preparación del bloque. Para ser específicos, supongamos que la entrada JPEG es una imagen RGB de 640 × 480 con 24 bits/píxel, como se muestra en la figura 7-72(a). Puesto que el uso de luminancia y crominancia da una mejor compresión, primero calcularemos la luminancia, *Y*, y las dos crominancias, *I* y *Q* (para NTSC), de acuerdo con las siguientes fórmulas:

$$Y = 0.30R + 0.59G + 0.11B$$
$$I = 0.60R - 0.28G - 0.32B$$
$$Q = 0.21R - 0.52G + 0.31B$$

En PAL, las crominancias se llaman *U* y *V*, y los coeficientes son diferentes, pero la idea es la misma. SECAM es diferente tanto de NTSC como de PAL.

Se construyen matrices separadas para *Y*, *I* y *Q*, cada una con elementos en el intervalo de 0 a 255. A continuación se promedian tramas de cuatro píxeles en las matrices *I* y *Q* para reducirlos a 320 × 240. Esta reducción tiene pérdidas, pero el ojo apenas lo nota, ya que responde a la luminancia más que a la crominancia; no obstante, comprime los datos en un factor de dos. Ahora se resta 128 a cada elemento de las tres matrices para poner el 0 a la mitad de la gama. Por último,

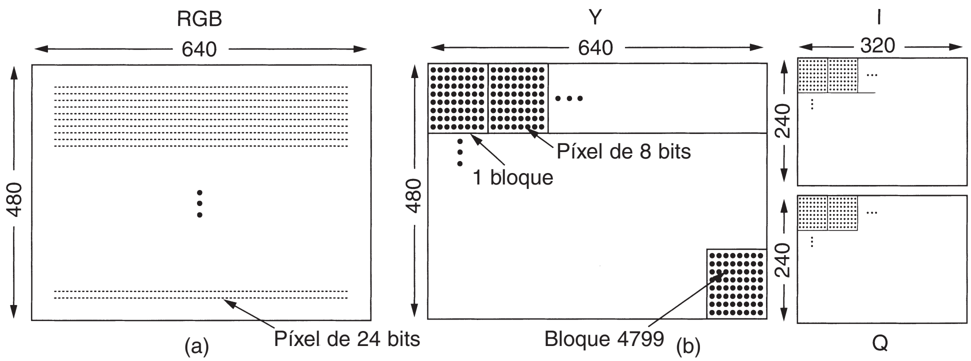


Figura 7-72. (a) Datos de entrada RGB. (b) Tras la preparación de bloques.

cada matriz se divide en bloques de 8 × 8. La matriz *Y* tiene 4800 bloques; las otras dos tienen 1200 bloques cada una, como se muestra en la figura 7-72(b).

El paso 2 de JPEG es aplicar de manera individual una **DCT** (**transformación por coseno discreto**) a cada uno de los 7200 bloques. La salida de cada DCT es una matriz de 8 × 8 de coeficientes DCT. El elemento DCT (0, 0) es el valor promedio del bloque. Los otros elementos indican la cantidad de potencia espectral que hay en cada frecuencia espacial. En teoría, una DCT no tiene pérdidas pero, en la práctica, el uso de números de punto flotante y funciones trascendentales siempre introducen algún error de redondeo que resulta en una pequeña pérdida de información. Estos elementos normalmente decaen con rapidez al alejarse del origen (0, 0), como se sugiere en la figura 7-73.

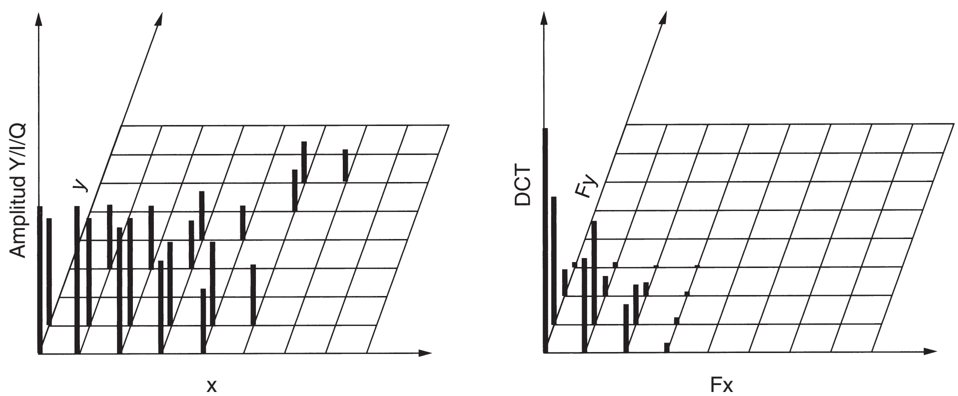


Figura 7-73. (a) Un bloque de la matriz *Y*. (b) Coeficientes DCT.

Una vez que la DCT está completa, el JPEG sigue con el paso 3, llamado **cuantización,** en el que se eliminan los coeficientes DCT menos importantes. Esta transformación (con pérdida) se hace dividiendo cada uno de los coeficientes de la matriz DCT de 8 × 8 entre un peso tomado de

una tabla. Si todos los pesos son 1, la transformación no hace nada. Sin embargo, si los pesos aumentan de manera considerable desde el origen, las frecuencias espaciales más altas se descartarán con rapidez.

En la figura 7-74 se da un ejemplo de este paso. Aquí vemos la matriz DCT inicial, la tabla de cuantización y el resultado obtenido al dividir cada elemento DCT entre el elemento correspondiente de la tabla de cuantización. Los valores de ésta no son parte del estándar JPEG. Cada aplicación debe proporcionar sus propios valores, lo que le permite controlar el equilibrio pérdida-compresión.

Coeficientes DCT

150	80	40	14	4	2	1	0
92	75	36	10	6	1	0	0
52	38	26	8	7	4	0	0
12	8	6	4	2	1	0	0
4	3	2	0	0	0	0	0
2	2	1	1	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Tabla de cuantización

1	1	2	4	8	16	32	64
1	1	2	4	8	16	32	64
2	2	2	4	8	16	32	64
4	4	4	4	8	16	32	64
8	8	8	8	8	16	32	64
16	16	16	16	16	16	32	64
32	32	32	32	32	32	32	64
64	64	64	64	64	64	64	64

Coeficientes cuantizados

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figura 7-74. Cálculo de los coeficientes DCT cuantizados.

El paso 4 reduce el valor (0, 0) de cada bloque (el de la esquina superior izquierda) reemplazándolo por el valor de su diferencia respecto al elemento correspondiente del bloque previo. Puesto que estos elementos son las medias de sus respectivos bloques, deben cambiar lentamente, por lo que al tomarse sus valores diferenciales se debe reducir la mayoría de ellos a valores pequeños. No se calculan diferenciales de los otros valores. Los valores (0, 0) se conocen como componentes de CD; los otros valores son los componentes de CA.

El paso 5 hace lineales los 64 elementos y aplica codificación por longitud de serie a la lista. Barrer el bloque de izquierda a derecha y luego de arriba abajo no concentra los ceros, por lo que se usa un patrón de barrido de zigzag, como se muestra en la figura 7-75. En este ejemplo, el patrón de zigzag produce 38 ceros consecutivos al final de la matriz. Esta cadena puede reducirse a una sola cuenta diciendo que hay 38 ceros, una técnica conocida como **codificación por longitud de serie.**

Ahora tenemos una lista de números que representan la imagen (en espacio de transformación). El paso 6 aplica codificación de Huffman a los números para su almacenamiento o transmisión, asignando códigos más cortos de números comunes que de no comunes.

JPEG puede parecer complicado, pero eso es porque *es* complicado. Aun así se usa ampliamente, debido a que con frecuencia produce una compresión de 20:1 o mejor. La decodificación de una imagen JPEG requiere la ejecución hacia atrás del algoritmo. JEPG es más o menos simétrico: la decodificación tarda tanto como la codificación. Esta propiedad no se aplica a todos los algoritmos de compresión, como veremos a continuación.

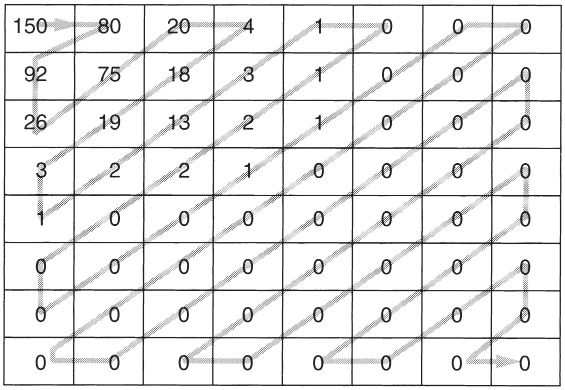


Figura 7-75. Orden en que se transmiten los valores cuantizados.

Estándar MPEG

Por último, llegamos al corazón del asunto: los estándares **MPEG** (**Grupo de Expertos en Imágenes en Movimiento). É**stos son los algoritmos principales usados para comprimir vídeo y han sido estándares internacionales desde 1993. Puesto que las películas contienen imágenes y sonido, MPEG puede comprimir tanto audio como vídeo. Ya examinamos la compresión de audio y la de imágenes, por lo que ahora nos enfocaremos principalmente en la compresión de vídeo.

El primer estándar terminado fue el MPEG-1 (Estándar Internacional 11172); su meta fue producir salida con calidad de videograbadora (352 × 240 para NTSC) usando una tasa de bits de 1.2 Mbps. Una imagen de 352 × 240 con 24 bits/píxeles y 25 tramas/seg requiere 50.7 Mbps, por lo que reducirla a 1.2 Mbps no es nada fácil. Se necesita un factor de compresión 40. MPEG-1 puede transmitirse sobre líneas de transmisión de par trenzado a distancias modestas. MPEG-1 también se usa para almacenar películas en CD-ROM.

El siguiente estándar de la familia MPEG fue MPEG-2 (Estándar Internacional 13818), que originalmente se diseñó para comprimir vídeo con calidad de difusión a 4 o 6 Mbps, a fin de que se ajustara en un canal de difusión NTSC o PAL. Posteriormente, MPEG-2 se extendió para soportar resoluciones mayores, entre ellas HDTV. En la actualidad es muy común y forma la base para el DVD y la televisión por satélite digital.

Los principios básicos de MPEG-1 y MPEG-2 son parecidos, pero los detalles son diferentes. A primera vista, MPEG-2 es un supergrupo del MPEG-1, con características adicionales, formatos de trama y opciones de codificación. Estudiaremos MPEG-1 primero y MPEG-2 después.

MPEG-1 tiene tres partes: audio, vídeo y sistema, que integra los otros dos, como se muestra en la figura 7-76. Los codificadores de audio y vídeo funcionan en forma independiente, lo que hace surgir la cuestión de cómo se sincronizan los dos flujos en el receptor. Este problema se resuelve teniendo un reloj de sistema de 90 kHz que proporciona el valor de tiempo actual a ambos codificadores. Estos valores son de 33 bits, para permitir que las películas duren 24 horas sin que

den la vuelta. Estas marcas de tiempo se incluyen en la salida codificada y se propagan hasta el receptor, que puede usarlas para sincronizar los flujos de audio y vídeo.

Ahora consideremos la compresión del vídeo MPEG-1. Existen dos clases de redundancia en las películas: espacial y temporal. MPEG-1 aprovecha ambas. La redundancia espacial puede utilizarse simplemente codificando por separado cada trama mediante JPEG. Este enfoque se usa en forma ocasional, sobre todo cuando se requiere acceso aleatorio a cada trama, como en la edición de producciones de vídeo. En este modo, se puede lograr un ancho de banda comprimido en el rango de 8 a 10 Mbps.

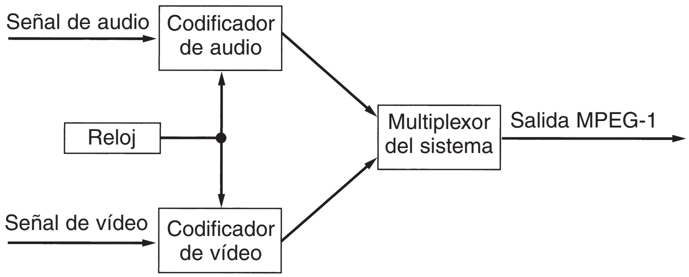


Figura 7-76. Sincronización de los flujos de audio y vídeo en MPEG-1.

Puede lograrse compresión adicional aprovechando el hecho de que las tramas consecutivas a menudo son casi idénticas. Este efecto es menor de lo que podría pensarse, puesto que muchos cineastas hacen cortes entre escenas cada 3 o 4 segundos (tome el tiempo de una película y cuente las escenas). No obstante, incluso una serie de 75 tramas muy parecidas ofrece una reducción importante que simplemente codificar cada trama por separado mediante JPEG.

En escenas en las que la cámara y el fondo son estacionarios y uno o dos actores se mueven con lentitud, casi todos los píxeles serán idénticos de una trama a otra. Aquí bastará con restar cada trama de la anterior y con aplicar JPEG a la diferencia. Sin embargo, esto no es muy bueno en las escenas en las que la cámara hace una panorámica o un acercamiento. Lo que se necesita es una forma para compensar este movimiento. Esto es precisamente lo que hace MPEG, y es la diferencia principal entre MPEG y JPEG.

La salida de MPEG-1 consiste en cuatro tipos de trama:

1. Tramas I (intracodificadas): imágenes fijas autocontenidas codificadas en JPEG.
2. Tramas P (predictivas): diferencia de bloque por bloque con la trama anterior.
3. Tramas B (bidireccionales): diferencias entre la trama anterior y la siguiente.
4. Tramas D (codificación CD): promedios de bloque usados para avance rápido.

Las tramas I son sólo imágenes fijas codificadas con una variante de JPEG, y también con luminancia de definición completa y crominancia de definición media sobre cada eje. Es necesario hacer que las tramas I aparezcan en forma periódica en el flujo de salida por tres razones. Primera, MPEG-1 puede usarse para transmisión de multidifusión, en la que los usuarios pueden realizar la sintonización a voluntad. Si todas las tramas dependieran de sus antecesores remontándose a la primera trama, cualquiera que no recibiera la primera trama nunca podría decodificar las tramas subsiguientes. Segunda, si una trama se recibiera con error, no sería posible ninguna decodificación posterior. Tercera, sin tramas I, al hacer un avance o retroceso rápido, el decodificador tendría que calcular cada trama por la que pasa para conocer el valor completo de aquella en la que se detiene. Por estas tres razones, se insertan tramas I en la salida una o dos veces por segundo.

En contraste, las tramas P codifican las diferencias entre tramas; se basan en la idea de los **macrobloques,** que cubren 16×16 píxeles de espacio de luminancia y 8×8 píxeles de espacio de crominancia. Un macrobloque se codifica buscando en la trama previa algo igual a él o ligeramente diferente de él.

En la figura 7-77 se muestra un caso en el que serían útiles las tramas P. Aquí vemos tres tramas consecutivas que tienen el mismo fondo, pero en las que cambia la posición de una persona. Los macrobloques que contienen el fondo serán exactamente iguales, pero los macrobloques que contienen la persona estarán desfasados en alguna cantidad desconocida y tendrán que rastrearse.

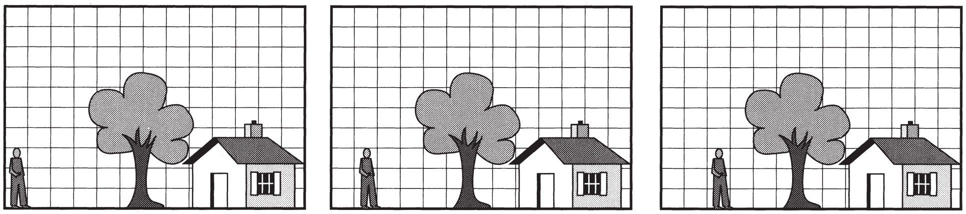


Figura 7-77. Tres tramas consecutivas.

El estándar MPEG-1 no especifica cómo efectuar la búsqueda, ni qué tan profunda o buena debe ser. Éstas son decisiones que dependen de cada implementación. Por ejemplo, una implementación podría buscar un macrobloque en la posición actual, pero de la trama anterior, y con todas las demás posiciones desplazadas $\pm\Delta x$ en la dirección x y $\pm\Delta y$ en la dirección y. Para cada posición, podría calcularse la cantidad de equivalencias en la matriz de luminancia. La posición con el puntaje más alto se declararía ganadora, siempre y cuando estuviera por encima de un umbral predefinido. En caso contrario, se diría que falta el macrobloque. Por supuesto, pueden usarse algoritmos mucho más refinados.

Si se encuentra un macrobloque, se codifica tomando la diferencia respecto a su valor en la trama previa (para la luminancia y ambas crominancias). Estas matrices de diferencias son el objeto de la transformación por coseno discreto, cuantización, codificación por longitud de serie y

codificación Huffman, al igual que en el JPEG. El valor del macrobloque en el flujo de salida es entonces el vector de movimiento (la distancia que se movió el macrobloque de su posición previa en cada sentido), seguido de la lista de codificación Huffman de los números. Si el macrobloque no se encuentra en la trama previa, se codifica el valor actual con JPEG, igual que en una trama I.

Es claro que este algoritmo es altamente asimétrico. Una implementación es libre de intentar todas las posiciones de la trama previa si lo desea, en un intento desesperado por localizar todos los macrobloques previos. Este método reducirá al mínimo el flujo MPEG-1 codificado al costo de una codificación muy lenta. Este método podría estar bien para la codificación de una sola vez de una cineteca, pero sería terrible para las videoconferencias en tiempo real.

De la misma manera, cada implementación puede decidir lo que constituye un macrobloque “encontrado”. Esto permite que los implementadores compitan según la calidad y velocidad de sus algoritmos, pero siempre produce MPEG-1 que se apega. Sea cual sea el algoritmo de búsqueda usado, la salida final es la codificación JPEG del macrobloque actual, o la codificación JPEG de la diferencia entre el macrobloque actual y el de la trama previa, con un desplazamiento especificado respecto a la trama actual.

Hasta ahora, la decodificación de MPEG-1 es directa. La decodificación de tramas I es igual a la de las imágenes JPEG. La decodificación de tramas P requiere que el decodificador almacene en el búfer la trama previa, y luego construya la nueva en un segundo búfer con base en macrobloques completamente codificados y en macrobloques que contienen diferencias respecto a la trama previa. La nueva trama se ensambla macrobloque por macrobloque.

Las tramas B son parecidas a las P, excepto que permiten que el macrobloque de referencia esté en una trama previa o en una trama posterior. Esta libertad adicional permite mejorar la compensación del movimiento y también es útil cuando pasan objetos por delante o detrás de otros objetos. Para ejecutar la codificación de tramas B, el codificador necesita tener a la vez tres tramas decodificadas en la memoria: la anterior, la actual y la siguiente. Aunque las tramas B producen la mejor compresión, no todas las implementaciones las soportan.

Las tramas D sólo se usan para visualizar una imagen de baja definición cuando se realiza un rebobinado o un avance rápido. Hacer la decodificación MPEG-1 normal en tiempo real ya es bastante difícil. Esperar que el decodificador lo haga mientras trabaja a diez veces su velocidad normal es pedir demasiado. En su lugar, las tramas D se utilizan para producir imágenes de baja resolución. Cada entrada de trama D simplemente es el valor promedio de un bloque, sin mayor codificación, lo que simplifica la presentación en tiempo real. Este mecanismo es importante para permitir que la gente barra un vídeo a alta velocidad en busca de una escena en particular. Por lo general, las tramas D se colocan justo antes de las tramas I correspondientes, por lo que si se detiene el avance rápido, será posible ver nuevamente a una velocidad normal.

Una vez terminado nuestro tratamiento de MPEG-1, pasemos al de MPEG-2. La codificación MPEG-2 es parecida en lo fundamental a la codificación MPEG-1, con tramas I, tramas P y tramas B. Sin embargo, no se soportan las tramas D. Además, la transformación por coseno discreto es de 10×10 en lugar de 8×8, lo que da 50% más de coeficientes y, por lo tanto, mayor calidad. Puesto que

MPEG-2 está dirigido a la televisión difundida al igual que al DVD, reconoce imágenes tanto progresivas como entrelazadas, mientras que MPEG-1 reconoce sólo las imágenes progresivas. También son diferentes otros detalles menores entre los dos estándares.

En lugar de soportar sólo un nivel de resolución, MPEG-2 soporta cuatro: bajo (352×240), medio (720×480), alto 1440 (1440×1152) y alto (1920×1080). La resolución baja es para las VCRs y por compatibilidad hacia atrás con MPEG-1. La resolución media es la normal para la difusión NTSC. Las otras dos son para HDTV. Para una salida de alta calidad, MPEG-2 por lo general se ejecuta de 4 a 8 Mbps.

7.4.8 Vídeo bajo demanda

El vídeo bajo demanda a veces se compara con una tienda de renta de vídeos. El usuario (cliente) selecciona cualquiera de los vídeos disponibles y se lo lleva a casa para verlo. Con el vídeo bajo demanda, la selección se realiza en casa utilizando el control remoto de la televisión y el vídeo comienza de inmediato. No se necesita ningún viaje a la tienda. Sobra decir que la implementación del vídeo bajo demanda es un tanto más complicada que su descripción. En esta sección daremos un repaso general de los conceptos básicos y de su implementación.

¿El vídeo bajo demanda es en realidad como rentar un vídeo, o se parece más a seleccionar una película de un sistema de cable de 500 canales? La respuesta tiene implicaciones técnicas importantes. En particular, los usuarios que rentan vídeos están acostumbrados a la idea de poder detener el vídeo, hacer un viaje rápido a la cocina o al baño, y luego continuar a partir de donde detuvieron el vídeo. Los televidentes no esperan tener la capacidad de poner en pausa los programas.

Si el vídeo bajo demanda va a competir con éxito contra las tiendas de renta de vídeos, podría ser necesario dar a los usuarios la capacidad de detener, iniciar y rebobinar los vídeos a voluntad. Para ello, el proveedor de vídeos tendrá que transmitir una copia individual a cada quien.

Por otra parte, si el vídeo bajo demanda se considera más como una televisión avanzada, entonces puede ser suficiente hacer que el proveedor de vídeo inicie cada vídeo popular, digamos, cada 10 minutos, y luego lo exhiba sin parar. Un usuario que desee ver un vídeo popular podría tener que esperar hasta 10 minutos para que comience. Aunque no es posible la pausa/reinicio aquí, un televidente que regrese a la sala tras una interrupción corta puede pasar a otro canal que muestre el mismo vídeo, pero con 10 minutos de retraso. Una parte del material se repetirá, pero no se perderá nada. Este esquema se llama **casi vídeo bajo demanda,** y ofrece la posibilidad de un costo mucho menor, puesto que la misma alimentación del servidor de vídeo puede llegar a muchos usuarios al mismo tiempo. La diferencia entre vídeo bajo demanda y casi vídeo bajo demanda es parecida a la diferencia entre manejar su propio auto y tomar el autobús.

Ver películas (casi) bajo demanda es un servicio de una gran grama de nuevos servicios que podrían hacerse realidad una vez que estén disponibles las redes de banda amplia. En la figura 7-78 se muestra el modelo general utilizado por mucha gente. Aquí vemos una red dorsal de área

amplia (nacional o internacional) de alto ancho de banda como centro del sistema. Conectadas a ella hay miles de redes de distribución locales, como TV por cable o sistemas de distribución de compañías telefónicas. Los sistemas de distribución locales llegan hasta las casas de la gente, donde terminan en **cajas de control,** que de hecho son potentes computadoras personales especializadas.

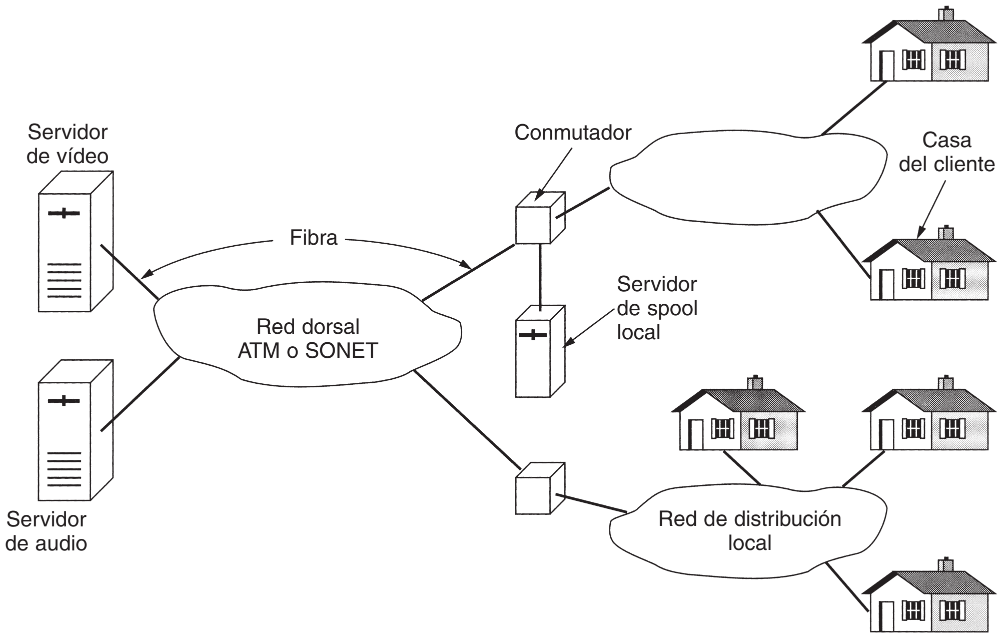


Figura 7-78. Esquema general de un sistema de vídeo bajo demanda.

Hay miles de proveedores de información conectados a la red dorsal mediante fibras ópticas de alto ancho de banda. Algunos de éstos ofrecerán vídeo o audio de pago por evento. Otros ofrecerán servicios especializados, como compras desde casa (en los que se permite que el televidente gire una lata de sopa y haga un acercamiento a la lista de ingredientes que aparece en la etiqueta de dicha lata, o que vea un fragmento de vídeo que trata sobre cómo manejar una podadora que utiliza gasolina). Sin duda, pronto habrá la disponibilidad de deportes, noticias, capítulos viejos del "Chapulín Colorado", acceso a WWW y otras innumerables posibilidades.

En el sistema también se incluyen servidores de *spool* de E/S locales que permitirán acercar los vídeos al usuario, a fin de ahorrar ancho de banda durante las horas pico. La manera en que encajarán estas partes y quién será el dueño de qué son temas de intenso debate en la industria. A continuación examinaremos el diseño de las partes principales del sistema: los servidores de vídeo y la red de distribución.

Servidores de vídeo

Para tener (casi) vídeo bajo demanda, necesitamos **servidores de vídeo** capaces de almacenar y sacar una gran cantidad de películas de manera simultánea. La cantidad total de películas que se han filmado se estima en 65,000 (Minoli, 1995). Cuando se comprime con MPEG-2, una película normal ocupa unos 4 GB, por lo que 65,000 de ellas requerirán unos 260 terabytes. Sume a todo esto todos los programas de televisión, filmaciones de deportes, noticieros, catálogos parlantes de compras, etcétera, y queda claro que tenemos en nuestras manos un problema de almacenamiento de proporciones mayúsculas.

La manera más sencilla de almacenar volúmenes grandes de información es en cinta magnética. Siempre ha sido el caso y probablemente seguirá siéndolo. Una cinta Ultrium puede almacenar 200 GB (50 películas) a un costo de aproximadamente 1-2 dólares/película. En el mercado hay servidores mecánicos grandes que almacenan miles de cintas y tienen un brazo de robot para traer cualquier cinta e introducirla en la unidad de cinta. El problema de estos sistemas es el tiempo de acceso (especialmente para la película 50 de una cinta), la tasa de transferencia y la cantidad limitada de unidades de cinta (para proporcionar *n* películas a la vez, se requieren *n* unidades).

Por fortuna, la experiencia de las tiendas de renta de vídeos, bibliotecas públicas y otras organizaciones similares demuestra que no todos los productos tienen la misma popularidad. Experimentalmente, cuando hay *N* películas disponibles, la fracción de todas las solicitudes que pide el elemento número *k* en popularidad es de aproximadamente C/k. Aquí, *C* se calcula para normalizar la suma a 1, es decir

$$C = 1/(1 + 1/2 + 1/3 + 1/4 + 1/5 + \cdots + 1/N)$$

Por lo tanto, la película más vista es siete veces más popular que la película número siete. Este resultado se conoce como **ley de Zipf** (Zipf, 1949).

El hecho de que algunas películas sean mucho más populares que otras sugiere una solución posible en forma de jerarquía de almacenamiento, como se muestra en la figura 7-79. Aquí, el desempeño aumenta a medida que se sube en la jerarquía.

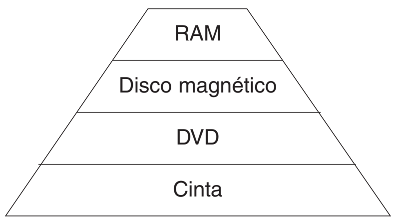


Figura 7-79. Jerarquía de almacenamiento de un servidor de vídeo.

Una alternativa a la cinta es el almacenamiento óptico. Los DVDs actuales almacenan 4.7 GB, que sirve para una película, pero la siguiente generación podrá almacenar dos películas. Aunque los tiempos de búsqueda son lentos en comparación con los discos magnéticos (50 mseg contra

5 mseg), su bajo costo y alta confiabilidad hace que los tocadiscos ópticos que contienen miles de DVDs sean una buena alternativa a la cinta para las películas de mayor uso.

A continuación están los discos magnéticos. Éstos tienen tiempos de acceso cortos (5 mseg), tasas de transferencia altas (320 MB/seg para SCSI 320) y capacidades sustanciales (> 100 GB), lo que los hace adecuados para guardar películas que en realidad se están transmitiendo (a diferencia de simplemente estar almacenadas en caso de que alguien quiera verlas). Su desventaja principal es el alto costo de almacenar películas a las que pocas veces se accede.

En la parte superior de la pirámide de la figura 7-79 está la RAM. Ésta es el medio de almacenamiento más rápido, pero también el más costoso. Cuando los precios de la RAM bajen a $50/GB, la RAM de una película de 4 GB costará $200, por lo que la RAM de 100 películas costará $20,000, debido a que éstas ocuparán 400 GB. Sin embargo, se está volviendo algo factible el que un servidor de vídeo que almacene 100 películas pueda mantenerlas a todas en la RAM. Y si dicho servidor tiene 100 clientes, pero éstos ven colectivamente sólo 20 películas diferentes, no sólo es algo factible sino un buen diseño.

Debido a que un servidor de vídeo en realidad es un dispositivo masivo de E/S en tiempo real, necesita una arquitectura de hardware y software diferente de la de una PC o una estación de trabajo UNIX. En la figura 7-80 se ilustra la arquitectura del hardware de un servidor de vídeo típico. El servidor tiene una o más CPUs de alto desempeño, cada una con un poco de memoria local, una memoria principal compartida, un caché de RAM masivo para películas populares, una variedad de dispositivos de almacenamiento para guardar las películas, y algún hardware de red, normalmente una interfaz óptica con una red dorsal ATM (o SONET) a OC-12 o mayor. Estos subsistemas se conectan mediante un bus de velocidad extremadamente alta (cuando menos 1 GB/seg).

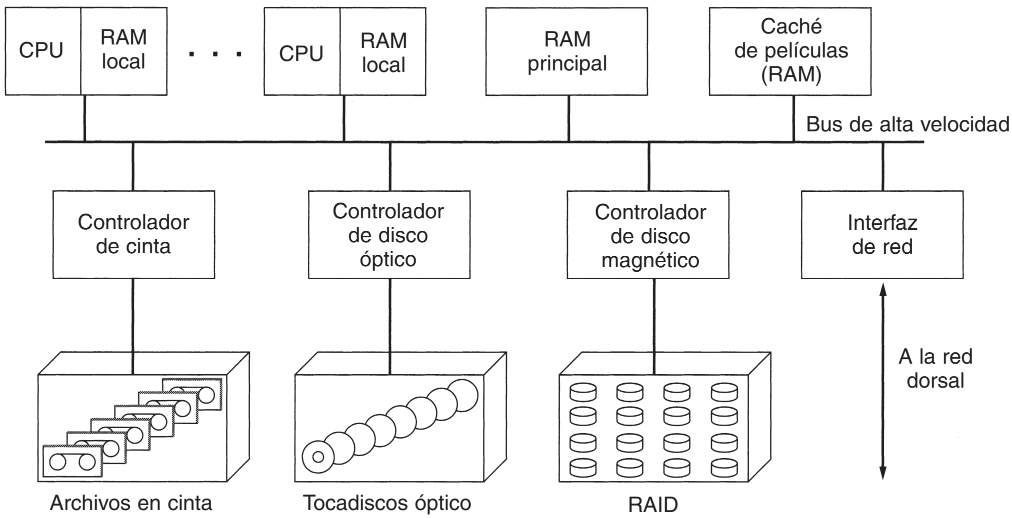


Figura 7-80. Arquitectura del hardware de un servidor de vídeo típico.

Ahora veamos brevemente el software de un servidor de vídeo. Las CPUs se usan para aceptar solicitudes de los usuarios, localizar películas, mover datos entre dispositivos, facturar al cliente

y muchas funciones más. Algunas de éstas no son de sincronización crítica, pero muchas otras sí, por lo que algunas de las CPUs, si no es que todas, tendrán que ejecutar un sistema operativo en tiempo real, como un micronúcleo en tiempo real. Por lo general, estos sistemas dividen el trabajo en tareas pequeñas, con un plazo de terminación conocido. Después el planificador puede ejecutar un algoritmo del tipo del siguiente plazo de terminación más cercano o el algoritmo de tasa monotónica (Liu y Layland, 1973).

El software de la CPU también define la naturaleza de la interfaz que el servidor presenta a los clientes (servidores de *spool* o cajas de control colocadas sobre el televisor). Son comunes dos diseños. El primero es un sistema tradicional de archivos, en el que los clientes pueden abrir, leer, escribir y cerrar archivos. Aparte de las complicaciones generadas por la jerarquía de almacenamiento y las consideraciones en tiempo real, uno de tales servidores puede tener un sistema de archivos modelado según el de UNIX.

El segundo tipo de interfaz se basa en el modelo de la videograbadora. Los comandos le piden al servidor que abra, ejecute, haga pausa, avance rápido y rebobine archivos. La diferencia con el modelo UNIX es que, una vez que se ha emitido un comando *PLAY*, el servidor simplemente sigue sacando datos a velocidad constante, sin que se requieran comandos nuevos.

El corazón del software del servidor de vídeo es el software de administración de disco, que tiene dos tareas principales: colocar películas en el disco magnético cuando tienen que traerse de almacenamiento óptico o cinta, y manejar solicitudes de disco para los muchos flujos de salida. La colocación de las películas es importante, puesto que puede afectar en gran medida el desempeño.

Dos maneras posibles de organizar el almacenamiento en disco son la granja de discos y el arreglo de discos. En la **granja de discos,** cada unidad contiene algunas películas completas. Por cuestiones de desempeño y confiabilidad, cada película debe estar presente en cuando menos dos unidades, tal vez más. La otra organización del almacenamiento es el **arreglo de discos** o **RAID** (**arreglo redundante de discos baratos**), en el que cada película se divide entre varias unidades, por ejemplo, el bloque 0 en la unidad 0, el bloque 1 en la unidad 1, etcétera, con el bloque $n - 1$ en la unidad $n - 1$. Después de eso, el ciclo se repite, con el bloque n en la unidad 0, etcétera. Esta organización se llama **división** (***striping***).

Un arreglo de discos con división tiene varias ventajas respecto a una granja de discos. Primero, las n unidades pueden operar en paralelo, aumentando el desempeño en un factor de n. Segundo, el arreglo puede hacerse redundante agregando una unidad extra a cada grupo de n, donde la unidad redundante contiene el OR exclusivo bloque por bloque de las otras unidades, a fin de permitir la recuperación completa de datos en caso de que una unidad falle. Por último, se resuelve el problema del equilibrio de la carga (no se requiere colocación manual para evitar que todas las películas populares se encuentren en la misma unidad). Por otra parte, la organización por arreglo de discos es más complicada que la granja de discos y es altamente sensible a diversas fallas; también es poco adecuada para operaciones de videograbadora, por ejemplo, para el rebobinado y avance rápido de una película.

La otra tarea del software del disco es dar servicio a todos los flujos de salida en tiempo real y cumplir sus restricciones de temporización. Hace apenas algunos años, esto requería algoritmos

de programación de disco complejos, pero ahora que los precios de la memoria están tan bajos, es posible un método mucho más simple. Por cada flujo servido, se almacena en RAM un búfer de, digamos, 10 seg de vídeo (5 MB). Se llena mediante un proceso de disco y se vacía mediante un proceso de red. Con 500 MB de RAM se pueden alimentar 100 flujos directamente de la RAM. Por supuesto, el subsistema de disco debe tener una tasa de datos sostenida de 50 MB/seg para mantener los búferes llenos, pero un RAID construido a partir de discos SCSI de alta calidad puede manejar fácilmente este requerimiento.

La red de distribución

La red de distribución es el grupo de conmutadores y líneas entre el origen y el destino. Como vimos en la figura 7-78, esta red consiste en una red dorsal conectada a la red local de distribución. Generalmente, la red dorsal es conmutada y la red de distribución local no lo es.

El requisito principal impuesto en la red dorsal es un ancho de banda alto. Una fluctuación baja también era un requisito, pero ahora ya no lo es, incluso ni en las PCs actuales más pequeñas, las cuales tienen la capacidad de almacenar en el búfer 10 seg de vídeo MPEG-2 de alta calidad.

La distribución local es caótica, pues las diferentes compañías usan redes distintas en diferentes regiones. Las compañías telefónicas, las de televisión por cable y los nuevos participantes, como las compañías de electricidad, están convencidos de que el que llegue primero será el gran ganador, por lo que ahora vemos una proliferación en la instalación de nuevas tecnologías. En Japón, algunas compañías de alcantarillado están en el negocio de Internet, argumentando que ellas tienen el canal más grande en la casa de todos (ejecutan una fibra óptica a través de él, pero deben ser muy cuidadosas sobre en qué lugar emerge). Los cuatro esquemas principales de distribución local de vídeo bajo demanda tienen las siglas ADSL, FTTC, FTTH y HFC. Ahora los explicaremos.

ADSL (**Línea Digital de Suscriptor Asimétrica**) fue el primer competidor de la industria telefónica por el premio de la distribución local. La idea es que en prácticamente cada casa de Estados Unidos, Europa y Japón ya tiene un par trenzado de cobre (para el servicio telefónico analógico). Si estos cables pudieran usarse para el vídeo bajo demanda, las compañías telefónicas podrían ganar.

Por supuesto, el problema es que estos cables no pueden manejar ni siquiera MPEG-1 a través de su recorrido normal de 10 km, y ni mencionar MPEG-2. El vídeo de movimiento y color completos, y de alta calidad necesita de 4 a 8 Mbps, dependiendo de la calidad deseada. ADSL realmente no es tan rápida, excepto para los ciclos locales muy cortos.

El segundo diseño de las compañías telefónicas es el **FTTC** (**Fibra Hasta la Acera**). En FTTC, la compañía telefónica tiende fibra óptica de la oficina central a cada barrio residencial, terminando en un dispositivo llamado **ONU** (**Unidad de Red Óptica**). En una ONU pueden terminar hasta 16 circuitos locales de cobre. Estos circuitos son lo bastante cortos para operar T1 o T2 de dúplex total en ellos, permitiendo películas MPEG-1 y MPEG-2, respectivamente. Además,

son posibles videoconferencias para trabajadores remotos y negocios pequeños, pues el FTTC es simétrico.

La tercera solución de las compañías telefónicas es tender fibra en todas las casas. Este sistema se llama **FTTH** (**Fibra Hasta la Casa**). En él, todo mundo puede tener una portadora OC-1, OC-3 o inclusive mayor, si se requiere. La FTTH es muy costosa y tardará años en hacerse realidad, pero ciertamente abrirá una vasta gama de nuevas posibilidades cuando finalmente ocurra. En la figura 7-63 vimos la forma en que cualquier persona puede operar su propia estación de radio. ¿Qué pensaría de que cada miembro de la familia opere su propia estación de televisión? ADSL, FTTC y FTTH son redes de distribución local de punto a punto, lo cual no es sorprendente debido a la forma en que está organizado el sistema telefónico actual.

Un enfoque completamente diferente es el **HFC** (**Híbrido Fibra/Coaxial**), la solución preferida que están instalando los proveedores de TV por cable y que se ilustra en la figura 2-47(a). La propuesta es la siguiente. Los cables coaxiales actuales de 300 a 450 MHz serán sustituidos por cables coaxiales de 750 MHz, elevando la capacidad de 50 a 75 canales de 6 MHz a 125 canales de 6 MHz. Setenta y cinco de los 125 canales se usarán para la transmisión de televisión analógica.

Los 50 canales nuevos se modularán de manera individual usando QAM-526, que proporciona unos 40 Mbps por canal, lo que da un total de 2 Gbps de ancho de banda nuevo. Los amplificadores *head-end* se moverán más hacia el interior de los vecindarios, de modo que cada cable pase por sólo 500 casas. Una división sencilla indica que es posible asignar a cada casa un canal dedicado de 4 Mbps, que puede manejar una película MPEG-2.

Aunque esto suena maravilloso, requiere que los proveedores de cable reemplacen todos los cables existentes con coaxial de 750 MHz, instalen nuevos amplificadores *head-end* y eliminen todos los amplificadores de una vía; en pocas palabras, que reemplacen todo el sistema de TV por cable. En consecuencia, la cantidad de infraestructura nueva es comparable con la que necesitan las compañías telefónicas para FTTC. En ambos casos, el proveedor local de la red debe tender fibra en los barrios residenciales. Nuevamente, en ambos casos, la fibra termina en un convertidor optoeléctrico. En FTTC, el segmento final es un circuito local punto a punto que usa cable de par trenzado. En HFC, el segmento final es cable coaxial compartido. Técnicamente, estos sistemas no son tan diferentes, como suelen asegurar sus patrocinadores.

No obstante, hay una diferencia real que es importante indicar. HFC usa un medio compartido sin conmutación ni enrutamiento. Cualquier información que se ponga en el cable puede ser retirada por cualquier suscriptor sin mayores trámites. FTTC, que es completamente conmutado, no tiene esta propiedad. Como resultado, los operadores HFC quieren que los servidores de vídeo transmitan flujos cifrados, de modo que los clientes que no hayan pagado una película no puedan verla. A los operadores FTTC no les interesa la encriptación, puesto que agrega complejidad, disminuye el desempeño y no proporciona seguridad adicional a su sistema. Desde el punto de vista de la compañía que opera un servidor de vídeo, ¿es buena idea encriptar? Un servidor operado por una compañía telefónica o una de sus subsidiarias o socios podría decidir de manera intencional no encriptar sus vídeos, e indicar como razón la eficiencia, pero en realidad lo hace para provocar pérdidas a sus competidores HFC.

Con todas estas redes locales de distribución es probable que se equipe a cada vecindario con uno o más servidores de *spool*. Éstos son, de hecho, simplemente versiones más pequeñas de los servidores de vídeo que estudiamos antes. La gran ventaja de estos servidores locales es que eliminan algo de la carga de la red dorsal.

Dichos servidores locales pueden recargarse con películas por reservación. Si las personas indican con suficiente anticipación al proveedor qué películas desean ver, éstas pueden bajarse al servidor local en horas no pico, con lo que se obtienen aún más ahorros. Probablemente esta observación llevará a los operadores de red a despedir a los ejecutivos de aviación que diseñan sus tarifas. De esta manera puede planearse un descuento en las tarifas de las películas ordenadas con 24 a 72 horas de anticipación que vayan a verse un martes o jueves por la tarde, antes de las 6 p.m. o después de las 11 p.m. Las películas ordenadas el primer domingo del mes antes de las 8 a.m. para verse un miércoles por la tarde de un día cuya fecha sea un número primo tendrán 43% de descuento, y así sucesivamente.

7.4.9 Mbone—Red dorsal de multidifusión

Mientras todas estas industrias hacen grandes (y muy publicitados) planes para el vídeo digital (inter)nacional bajo demanda, la comunidad de Internet ha estado implementando calladamente su propio sistema digital de multimedia, **MBone** (**Red Dorsal de Multidifusión).** En esta sección daremos un panorama de lo que es y cómo funciona.

Puede pensar en MBone como una radio y televisión de Internet. A diferencia del vídeo bajo demanda, donde lo más importante es solicitar y ver películas precomprimidas almacenadas en un servidor, MBone se usa para difundir audio y vídeo en vivo de forma digital por todo el mundo a través de Internet. MBone ha estado en operación desde comienzos de 1992. Se han difundido muchas conferencias científicas, especialmente las reuniones del IETF, así como eventos científicos de importancia noticiosa, como varios lanzamientos del trasbordador espacial. Alguna vez se difundió un concierto de los Rolling Stones por MBone, al igual que algunas partes del Festival de Cine de Canes. El hecho de que este suceso se califique como un hecho científico notable es discutible.

Desde el aspecto técnico, MBone es una red virtual sobrepuesta a Internet. Mbone consiste en islas capaces de multidifundir, conectadas mediante túneles, como se muestra en la figura 7-81. En esta figura, la MBone consiste en seis islas, *A* a *F*, conectadas mediante siete túneles. Cada isla (por lo común, una LAN o grupo de LANs interconectadas) soporta la multidifusión por *hardware* a sus *hosts*. Los túneles propagan paquetes MBone entre las islas. En el futuro, cuando todos los enrutadores sean capaces de manejar en forma directa tráfico de multidifusión, dejará de necesitarse esta superestructura, pero por el momento lleva a cabo el trabajo.

Cada isla contiene uno o más enrutadores especiales, llamados **enrutadores m** o *mrouters* (**enrutadores de multidifusión).** Algunos de éstos son en realidad enrutadores normales, pero otros sencillamente son estaciones de trabajo UNIX que ejecutan software especial de nivel de usuario (pero como la raíz). Los enrutadores m se conectan de manera lógica mediante túneles.

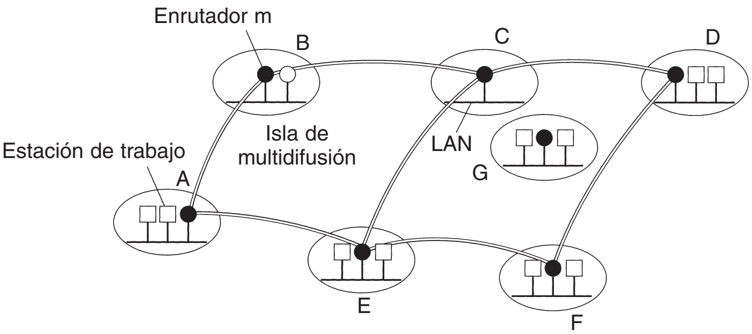


Figura 7-81. MBone consiste en islas de multidifusión conectadas mediante túneles.

Los paquetes MBone se encapsulan en paquetes IP y se envían como paquetes de unidifusión normales a la dirección IP del enrutador m de destino.

Los túneles se configuran de manera manual. Por lo general, un túnel corre por una ruta para la que existe una conexión física, pero esto no es un requisito. Si por accidente se cae la ruta física subyacente de un túnel, los enrutadores m que usan el túnel ni siquiera lo notarán, puesto que la Internet reenrutará automáticamente todo el tráfico IP entre ellos por otras líneas.

Cuando aparece una isla nueva que desea unirse a la MBone, como en el caso de *G* de la figura 7-81, su administrador envía un mensaje que indica su existencia a la lista de correo de MBone. Los administradores de los sitios cercanos se ponen en contacto con ella para acordar el establecimiento de túneles. A veces los túneles existentes se reordenan para aprovechar la nueva isla y optimizar la topología. A fin de cuentas, los túneles no existen físicamente, se definen mediante tablas en los enrutadores m y pueden agregarse, eliminarse o moverse con sólo cambiar estas tablas. Por lo general, cada país en la MBone tiene una red dorsal con islas regionales conectadas a ella. Normalmente, la MBone se configura con uno o dos túneles que cruzan el Océano Atlántico y el Pacífico, lo que hace que la MBone sea de escala global.

Por lo tanto, en cualquier momento la MBone consiste en una topología específica de túneles e islas, independiente de la cantidad de direcciones de multidifusión en uso en el momento y de quiénes las estén escuchando u observando. Esta situación es muy parecida a una subred normal (física), por lo que se aplican los mismos algoritmos de enrutamiento. En consecuencia, la MBone inicialmente usaba un algoritmo de enrutamiento, el **DVMRP** (**Protocolo de Enrutamiento Multidifusión de Vector de Distancia**)**,** que se basa en el algoritmo de vector de distancia Bellman-Ford. Por ejemplo, en la figura 7-81, la isla *C* puede enrutar a *A* vía *B* o *E* (o tal vez vía *D*). *C* lo decide tomando los valores que le dan los nodos sobre sus respectivas distancias a *A*, y sumando después su distancia a ellos. De esta manera, cada isla determina la mejor ruta a todas las demás islas. Sin embargo, las rutas en realidad no se usan de esta manera, como veremos en breve.

Consideremos ahora la manera en que ocurre la multidifusión. Para multidifundir un programa de audio o vídeo, una fuente primero debe adquirir una dirección de multidifusión clase D, que actúa como la frecuencia o el número de canal de una estación. Las direcciones clase D se reservan usando un programa que busca direcciones de multidifusión libres en una base de datos. Pueden ocurrir simultáneamente muchas multidifusiones, y un *host* puede "sintonizarse" con aquellas en la que está interesado si escucha en la dirección de multidifusión adecuada.

Periódicamente, cada enrutador m envía un paquete de difusión IGMP limitando a su isla, preguntando quién está interesado en qué canal. Los *hosts* que desean (continuar) recibiendo uno o más canales envían como respuesta otro paquete IGMP. Estas respuestas se reparten en el tiempo, para evitar sobrecargar la LAN local. Cada enrutador m mantiene una tabla de los canales que debe poner en sus LANs, para evitar el desperdicio de ancho de banda al multidifundir canales que nadie quiere.

Las multidifusiones se propagan por la MBone como se explica a continuación. Cuando una fuente de audio o vídeo genera un paquete nuevo, lo multidifunde a su isla local usando el recurso de multidifusión del hardware. El enrutador m local recoge este paquete, y después lo copia en todos los túneles a los que está conectado.

Cada enrutador m que recibe uno de tales paquetes por medio de un túnel lo revisa para ver si llegó por la mejor ruta, es decir, la ruta que, según su tabla, debe usarse para llegar al origen (como si fuera el destino). Si el paquete llegó por la mejor ruta, el enrutador m copia el paquete en todos sus otros túneles. Si el paquete llegó por una ruta subóptima, se descarta. Por ejemplo, en la figura 7-81, si las tablas de *C* le indican que use *B* para llegar a *A*, entonces cuando un paquete de multidifusión de *A* a *C* llega vía *B*, se copia de ahí a *D* y a *E*. Sin embargo, cuando un paquete de multidifusión de *A* a *C* llega vía *E* (que no es la mejor ruta), simplemente se descarta. Este algoritmo sólo es el reenvío por ruta invertida que vimos en el capítulo 5; aunque no es perfecto, es bastante bueno y muy sencillo de implementar.

Además de usar reenvío por ruta invertida para evitar inundar la Internet, también se usa el campo *Tiempo de vida* del IP para limitar el alcance de la multidifusión. Cada paquete comienza con algún valor (determinado por el origen). A cada túnel se le asigna un peso. Un paquete sólo pasa a través de un túnel si tiene suficiente peso; de otra manera, se descarta. Por ejemplo, los túneles transoceánicos se configuran normalmente con un peso de 128, por lo que los paquetes pueden limitarse al continente de origen dándoles un *Tiempo de vida* inicial de 127 o menos. Tras pasar por el túnel, al campo de *Tiempo de vida* se le resta el peso del túnel.

Aunque el algoritmo de enrutamiento de la MBone funciona, se han realizado muchas investigaciones para mejorarlo. Una propuesta mantiene la idea del enrutamiento por vector de distancia, pero hace jerárquico el algoritmo, agrupando en regiones las instalaciones de la MBone y enrutando primero hacia ellas (Thyagarajan y Deering, 1995).

Otra propuesta es usar una forma modificada del enrutamiento por estado de enlace en lugar del enrutamiento por vector de distancia. En particular, un grupo de trabajo del IETF está modificando el OSPF para adecuarlo a la multidifusión en un solo sistema autónomo. El OSPF de multidifusión resultante se llama **MOSPF** (Moy, 1994). Lo que hacen las modificaciones es que el mapa completo construido por el MOSPF lleva el registro de las islas y túneles de multidifusión, además de la información normal de enrutamiento. Si se conoce la topología completa, es directo

el cálculo de la mejor ruta desde cualquier isla a cualquier otra usando los túneles. Por ejemplo, puede usarse el algoritmo de Dijkstra.

Una segunda área de investigación es el enrutamiento entre los AS. Aquí, otro grupo de trabajo del IETF está desarrollando un algoritmo llamado **PIM** (**Multidifusión Independiente del Protocolo**). El PIM viene en dos versiones, dependiendo de si las islas son densas (casi todos quieren ver) o dispersas (casi nadie quiere ver). Ambas versiones usan las tablas de enrutamiento de unidifusión estándar, en lugar de crear una topología superpuesta como lo hacen DVMRP y MOSPF.

En el PIM-DM (modo denso), la idea es recortar las rutas inútiles. El recortado funciona como sigue. Cuando llega un paquete de multidifusión por el túnel "equivocado", se devuelve un paquete de recorte por dicho túnel, indicando al transmisor que cese el envío de paquetes desde la fuente en cuestión. Cuando un paquete llega por el túnel "correcto", se copia en todos los demás túneles que no han sido previamente recortados. Si todos los demás túneles se han recortado y no hay interés por el canal en la isla local, el enrutador m devuelve un mensaje de recorte por el canal "correcto". De esta manera, la multidifusión se adapta en forma automática y sólo va a donde se necesita.

El PIM-SM (modo disperso), que se describe en el RFC 2362, funciona de manera diferente. Aquí la idea es evitar la saturación de Internet porque tres personas de Berkeley quieren realizar una llamada en conferencia por una dirección clase D. El PIM-SM opera estableciendo puntos de reunión. Cada una de las fuentes de un grupo de multidifusión PIM disperso envía sus paquetes a los puntos de reunión. Cualquier sitio interesado en unirse solicita a uno de los puntos de reunión que establezca un túnel a él. De esta manera, todo el tráfico PIM-SM transporta por unidifusión en lugar de multidifusión. PIM-SM se está volviendo más popular, y MBone está migrando a su uso. A medida que el uso de PIM-SM se está volviendo más común, el de MOSPF está desapareciendo gradualmente. Por otro lado, la MBone misma parece algo estática y probablemente nunca se vuelva muy popular.

Sin embargo, la multimedia en red aún es un campo emocionante y en rápido movimiento, aunque MBone no sea un gran éxito. Las nuevas tecnologías y aplicaciones se anuncian a diario. La multidifusión y la calidad del servicio cada vez están más unidas, como se discute en (Striegel y Manimaran, 2002). Otro tema interesante es la multidifusión inalámbrica (Gossain y cols., 2002). Es probable que en los próximos años toda el área de la multidifusión y todo lo relacionado con ella siga siendo importante.

7.5 RESUMEN

Los nombres de Internet utilizan un esquema jerárquico llamado DNS. En el nivel superior se encuentran los dominios genéricos bien conocidos, entre ellos *com* y *edu*, así como cerca de 200 dominios de países. DNS está implementado como un sistema de base de datos distribuido con servidores en todo el mundo. DNS mantiene registros con direcciones IP, registros de correo y otra información. Al consultar un servidor DNS, un proceso puede asignar un nombre de dominio de Internet a la dirección IP utilizada para comunicarse con el dominio.

El correo electrónico es una de las dos aplicaciones principales para Internet. Todo mundo, desde niños pequeños hasta los abuelos, lo utiliza. La mayoría de los sistemas de correo electrónico en el mundo utiliza el sistema de correo definido en los RFCs 2821 y 2822. Los mensajes enviados en este sistema utilizan encabezados ASCII de sistema para definir propiedades de mensajes. MIME se puede utilizar para enviar mucho contenido de diversos tipos. Los mensajes se envían utilizando SMTP, que funciona al establecer una conexión TCP del *host* de origen al de destino y entregando de manera directa el correo electrónico a través de la conexión TCP.

La otra aplicación principal de Internet es World Wide Web. Web es un sistema para vincular documentos de hipertexto. Originalmente, cada documento era una página escrita en HTML con hipervínculos a otros documentos. Hoy en día, XML está comenzando gradualmente a tomar ventaja sobre HTML. Además, una gran cantidad de contenido se genera en forma dinámica utilizando secuencias de comandos en el servidor (PHP, JSP y ASP), así como secuencias de comandos en el cliente (principalmente JavaScript). Un navegador puede desplegar un documento estableciendo una conexión TCP con su servidor, solicitando el documento y después cerrando la conexión. Estos mensajes de solicitud contienen una variedad de encabezados para proporcionar información adicional. El almacenamiento en caché, la replicación y las redes de entrega de contenido se utilizan ampliamente para mejorar el desempeño de Web.

La Web inalámbrica apenas está comenzando. Los primeros sistemas son WAP e i-mode, cada uno con pantallas pequeñas y ancho de banda limitado, pero la siguiente generación será más poderosa.

La multimedia también es una estrella en ascenso en el firmamento de las redes. Permite que el audio y el vídeo sean digitalizados y transportados de manera electrónica para su despliegue. El audio requiere menos ancho de banda. El audio de flujo continuo, la radio en Internet y la voz sobre IP ahora son una realidad, y continuamente surgen nuevas aplicaciones. El vídeo bajo demanda es un área prometedora en la que hay un gran interés. Por último, la MBone es un servicio experimental de televisión digital mundial que se envía a través de Internet.

PROBLEMAS

1. Muchas computadoras de negocios tienen tres identificadores únicos en todo el mundo. ¿Cuáles son?

2. De acuerdo con la información que se dio en la figura 7-3, ¿*little-sister.cs.vu.nl* corresponde a una red A, B o C?

3. En la figura 7-3, ¿hay un punto después de *rowboat*? ¿Por qué no?

4. Adivine qué significa :-X (algunas veces se escribe como :-#).

5. DNS utiliza UDP en lugar de TCP. Si se pierde un paquete DNS, no hay recuperación automática. ¿Esto causa un problema, y si es así, cómo se resuelve?

6. Además de ser propensos a perderse, los paquetes UDP tienen una longitud máxima, potencialmente tan baja como 576 bytes. ¿Qué pasa cuando un nombre DNS que se va a buscar excede esta longitud? ¿Se puede enviar en dos paquetes?

7. ¿Una máquina con un solo nombre DNS puede tener múltiples direcciones IP? ¿Cómo puede ocurrir esto?

8. ¿Una computadora puede tener dos nombres DNS que pertenecen a dominios de nivel superior diferentes? De ser así, dé un ejemplo razonable. De lo contrario, explique por qué no.

9. El número de compañías con un sitio Web ha crecido de manera explosiva en los años recientes. Como resultado, miles de compañías están registradas con el dominio *com*, lo que causa una carga pesada en el servidor de nivel superior de este dominio. Sugiera una manera de aliviar este problema sin cambiar el esquema de nombres (es decir, sin introducir nuevos nombres de dominio de nivel superior). Es válido que su solución requiera cambios al código cliente.

10. Algunos sistemas de correo electrónico soportan un campo de encabezado *Content Return:*. Especifique si el cuerpo de un mensaje se va a regresar en caso de que no se entregue. ¿Este campo pertenece al sobre o al encabezado?

11. Los sistemas de correo electrónico necesitan directorios a fin de que se puedan buscar las direcciones de correo electrónico de las personas. Para construir tales directorios y para que la búsqueda sea posible, los nombres deben dividirse en componentes estándar (por ejemplo, nombre, apellido). Mencione algunos problemas que deben resolverse a fin de que un estándar mundial sea aceptable.

12. La dirección de correo electrónico de una persona es su nombre de inicio de sesión + @ + el nombre de un dominio DNS con un registro *MX*. Los nombres de inicio de sesión pueden ser nombres de pila, apellidos, iniciales y todos los tipos de nombres. Suponga que una compañía grande decidió que se estaba perdiendo mucho correo debido a que las personas no sabían el nombre de inicio de sesión del receptor. ¿Hay alguna forma de que dicha compañía arregle este problema sin cambiar el DNS? De ser así, dé una propuesta y explique cómo funciona. De lo contrario, explique por qué no es posible.

13. Un archivo binario tiene una longitud de 3072 bytes. ¿Qué longitud tendrá si se codifica mediante base64, y se inserta un par CR+LF después de cada 80 bytes enviados y al final?

14. Considere el esquema de codificación MIME entrecomillado-imprimible. Mencione un problema que no se analiza en el texto y proponga una solución.

15. Nombre cinco tipos MIME que no se listan en este libro. Puede verificar su navegador o Internet para obtener información.

16. Suponga que desea enviar un archivo MP3 a un amigo, pero el ISP de éste limita a 1 MB la cantidad de correo entrante y el archivo MP3 es de 4 MB. ¿Hay alguna forma para manejar esta situación utilizando el RFC 822 y MIME?

17. Suponga que alguien establece un demonio de vacaciones y que después envía un mensaje justo antes de terminar su sesión. Desgraciadamente, el receptor ha estado de vacaciones una semana y también tiene un demonio de vacaciones en su lugar. ¿Qué sucede a continuación? Las respuestas grabadas se enviarán y regresarán hasta que alguien regrese?

18. En cualquier estándar, como el RFC 822, se necesita una gramática precisa de lo que está permitido de manera que diferentes implementaciones puedan interactuar. Incluso los elementos simples se tienen que definir con cuidado. Los encabezados SMTP permiten espacios en blanco entre los *tokens*. Dé *dos* definiciones de alternativas razonables de espacios en blanco entre los *tokens*.

19. ¿El demonio de vacaciones es parte del agente de usuario o del agente de transferencia de mensajes? Por supuesto, se establece con el agente de usuario, pero ¿éste envía realmente las respuestas? Explique.

20. POP3 permite que los usuarios obtengan y bajen correo electrónico de un buzón remoto. ¿Esto significa que el formato interno de los buzones debe estandarizarse para que cualquier programa POP3 en el cliente pueda leer el buzón en cualquier servidor de correo? Explique su respuesta.

21. Desde el punto de vista de un ISP, POP3 e IMAP tienen diferencias importantes. Por lo general, los usuarios de POP3 vacían sus buzones todos los días. Los usuarios de IMAP mantienen su correo electrónico en el servidor de manera indefinida. Imagine que se le pide a usted que aconseje a un ISP sobre cuáles protocolos debe soportar. ¿Qué aspectos tomaría en cuenta?

22. ¿Webmail utiliza POP3, IMAP o ninguno? Si utiliza alguno de éstos, ¿por qué se eligió? Si no se utiliza ninguno, ¿cuál está más cerca de ser usado?

23. Cuando se envían las páginas Web, se les anteponen encabezados MIME. ¿Por qué?

24. ¿Cuándo son necesarios los visores externos? ¿Cómo sabe un navegador cuál utilizar?

25. ¿Es posible que cuando un usuario haga clic en un vínculo con Netscape se inicie una aplicación auxiliar en particular, y que cuando haga clic en el mismo vínculo en Internet Explorer se inicie una aplicación auxiliar completamente diferente, aunque el tipo MIME regresado en ambos casos sea idéntico? Explique su respuesta.

26. Un servidor Web de múltiples subprocesos está organizado como se muestra en la figura 7-21. Tarda 500 µseg en aceptar una solicitud y verificar el caché. La mitad del tiempo es para encontrar el archivo en el caché y para regresarlo de inmediato. En la otra mitad del tiempo, el módulo tiene que bloquearse por 9 mseg mientras su solicitud de disco se coloca en la cola y se procesa. ¿Cuántos módulos debe tener el servidor para mantener ocupada todo el tiempo a la CPU (suponiendo que el disco no es un cuello de botella)?

27. El URL *http* estándar da por hecho que el servidor Web está escuchando en el puerto 80. Sin embargo, es posible que un servidor Web escuche en otro puerto. Diseñe una sintaxis razonable para que un URL acceda a un archivo en un puerto no estándar.

28. Aunque no se mencionó en el texto, una forma alternativa de un URL es utilizar una dirección IP en lugar de su nombre DNS. Un ejemplo del uso de una dirección IP es *http://192.31.231.66/index.html*. ¿Cómo sabe el navegador si el nombre que sigue al esquema es un nombre DNS o una dirección IP?

29. Imagine que alguien del Departamento de Computación de Stanford acaba de escribir un nuevo programa que desea distribuir mediante FTP. Esa persona coloca el programa en el directorio *ftp/pub/freebies/-newprog.c* de FTP. ¿Cuál será el URL más probable de este programa?

30. En la figura 7-25, *www.aportal.com* mantiene un registro de las preferencias del usuario en una *cookie*. Una desventaja de este esquema es que las *cookies* están limitadas a sólo 4 KB, de manera que si las preferencias son grandes —por ejemplo, muchas acciones, equipos de deportes, tipos de noticias, el clima

de varias ciudades, ediciones especiales de varias categorías de productos, etcétera— puede alcanzarse el límite de 4 KB. Diseñe una forma alternativa para mantener el registro de las preferencias que no tengan este problema.

31. El Banco Sloth desea que sus clientes flojos puedan utilizar con facilidad su banca en línea, por lo que después de que un cliente firma y se autentica mediante una contraseña, el banco regresa una *cookie* que contiene un número de ID del cliente. De esta forma, el cliente no tiene que identificarse a sí mismo o escribir una contraseña en visitas futuras a la banca en línea. ¿Qué opina de esta idea? ¿Funcionará? ¿Es una buena idea?

32. En la figura 7-26, el parámetro *ALT* se establece en la etiqueta <img>. ¿Bajo qué condiciones lo utiliza el navegador, y cómo?

33. ¿Cómo utiliza HTML a fin de que se pueda hacer clic en una imagen? Dé un ejemplo.

34. Muestre la etiqueta <a> que se necesita para hacer que la cadena "ACM" sea un hipervínculo a *http://www.acm.org*.

35. Diseñe un formulario para que la nueva compañía Interburger permita ordenar hamburguesas a través de Internet. Dicho formulario debe incluir el nombre, la dirección y la ciudad del cliente, así como opciones que permitan seleccionar el tamaño (gigante o inmensa) y si llevará queso. Las hamburguesas se pagarán en efectivo a la entrega por lo que no se necesita información de tarjeta de crédito.

36. Diseñe un formulario que pida al usuario que teclee dos números. Cuando el usuario haga clic en el botón de envío, el servidor regresará la suma de dichos números. Escriba el servidor como una secuencia de comandos PHP.

37. Para cada una de las siguientes aplicaciones, indique si sería (1) posible y (2) mejor utilizar una secuencia de comandos PHP o una JavaScript y por qué.
(a) Desplegar un calendario por cada mes solicitado desde septiembre de 1752.
(b) Desplegar la agenda de vuelos de Ámsterdam a Nueva York.
(c) Graficar un polinomio a partir de coeficientes proporcionados por el usuario.

38. Escriba un programa en JavaScript que acepte un entero mayor que 2 e indique si es un número primo. Observe que JavaScript tiene instrucciones if y while con la misma sintaxis que C y Java. El operador de módulo es %. Si necesita la raíz cuadrada de *x*, utilice *Math.sqrt* (*x*).

39. Una página HTML es como sigue:

```
<html> <body>
<a href="www.info-source.com/welcome.html"> Haga clic aquí para obtener información </a>
</body> </html>
```

Si el usuario hace clic en el hipervínculo, se abre una conexión TCP y se envía una serie de líneas al servidor. Liste todas las líneas enviadas.

40. Es posible utilizar el encabezado *If-Modified-Since* para verificar si una página almacenada en caché aún es válida. Las solicitudes pueden realizarse para obtener páginas que contengan imágenes, sonido, vídeo, HTML, etcétera. ¿Cree que la efectividad de esta técnica es mejor o peor para imágenes JPEG en comparación con HTML? Piense con cuidado sobre lo que significa "efectividad" y explique su respuesta.

41. El día en que se celebra un evento deportivo importante, como el juego de campeonato de la NFL, muchas personas visitan el sitio Web oficial. ¿Es ésta una aglomeración instantánea en el mismo sentido que con las elecciones en Florida del 2000? ¿Por qué sí o por qué no?

42. ¿Tiene sentido que un solo ISP funcione como una CDN? De ser así, ¿cómo funcionaría? De lo contrario, ¿qué está mal con esa idea?

43. ¿Bajo qué condiciones es una mala idea utilizar una CDN?

44. Las terminales de los sistemas inalámbricos Web tienen un ancho de banda bajo, lo cual provoca que la codificación eficiente sea importante. Diseñe un esquema para transmitir texto en inglés de manera eficiente a través de un enlace inalámbrico a un dispositivo WAP. Puede asumir que la terminal tiene algunos megabytes de ROM y una CPU con potencia moderada. *Sugerencia:* piense en transmitir japonés, en el que cada símbolo es una palabra.

45. Un disco compacto contiene 650 MB de datos. ¿La compresión se utiliza para CDs de audio? Explique su respuesta.

46. En la figura 7-57(c) el ruido de cuantización ocurre debido al uso de muestras de 4 bits para representar nueve valores de señal. La primera muestra, en 0, es exacta, pero las siguientes no lo son. ¿Cuál es el porcentaje de error para las muestras en 1/32, 2/32 y 3/32 del periodo?

47. ¿Es posible utilizar un modelo psicoacústico para reducir el ancho de banda necesario para la telefonía de Internet? De ser así, ¿cuáles condiciones, si las hay, se tendrían que cumplir para que funcionara? De lo contrario, ¿por qué no es posible?

48. Un servidor de flujo continuo de audio tiene una distancia de una sola vía de 50 mseg con un reproductor de medios. Tiene una salida de 1 Mbps. Si el reproductor de medios tiene un búfer de 1 MB, ¿qué puede decir acerca de la posición de la marca de agua baja y de la alta?

49. El algoritmo de entrelazado de la figura 7-60 tiene la ventaja de que puede resolver la pérdida de un paquete ocasional sin introducir un hueco en la reproducción. Sin embargo, cuando se utiliza para la telefonía de Internet, también tiene una pequeña desventaja. ¿Cuál es?

50. ¿La voz sobre IP tiene los mismos problemas con los servidores de seguridad que el audio de flujo continuo? Explique su respuesta.

51. ¿Cuál es la tasa de bits para transmitir tramas de color sin comprimir de 800×600 píxeles con 8 bits/-píxel a 40 tramas/seg?

52. ¿Un error de 1 bit en una trama MPEG puede dañar más que la trama en la que ocurrió el error? Explique su respuesta.

53. Considere un servidor de vídeo de 100,000 clientes, en donde cada cliente ve dos películas por mes. La mitad de las películas se proporcionan a las 8 p.m. ¿Cuántas películas tiene que transmitir a la vez el servidor durante este periodo? Si cada película requiere 4 Mbps, ¿cuántas conexiones OC-12 necesita el servidor para la red?

54. Suponga que la ley de Zipf se cumple para acceder un servidor de vídeo con 10,000 películas. Si el servidor mantiene las 1000 películas más populares en un disco magnético y las restantes 9000 en un disco óptico, dé una expresión para la fracción de todas las referencias que estarán en un disco magnético. Escriba un pequeño programa para evaluar esta expresión de manera numérica.

55. Algunos cibernautas han registrado nombres de dominio que son muy parecidos a algunos sitios corporativos, por ejemplo, *www.microsfot.com,* con algunas diferencias en la ortografía. Haga una lista de por lo menos cinco dominios de este tipo.

56. Muchas personas han registrado nombres DNS que consisten en *www.palabra.com* donde *palabra* es una palabra común. Para cada una de las siguientes categorías, liste cinco sitios Web y resuma brevemente lo que es (por ejemplo, *www.estomago.com* es un gastroenterólogo de Long Island). Las categorías son: animales, comidas, objetos domésticos y partes del cuerpo. Para la última categoría, por favor apéguese a las partes del cuerpo que se encuentran arriba de la cintura.

57. Diseñe sus propios emoji utilizando un mapa de bits de 12×12. Incluya novio, novia, profesor y político.

58. Escriba un servidor POP3 que acepte los siguientes comandos: *USER*, *PASS*, *LIST*, *RETR*, *DELE* y *QUIT*.

59. Rescriba el servidor de la figura 6-6 como un servidor real Web utilizando el comando *GET* para HTTP 1.1. También debe aceptar el mensaje *Host*. El servidor debe mantener un caché de archivos recientemente obtenidos del disco y atender solicitudes del caché cuando sea posible.

8

SEGURIDAD EN REDES

Durante las primeras décadas de su existencia, las redes de computadoras fueron usadas principalmente por investigadores universitarios para el envío de correo electrónico, y por empleados corporativos para compartir impresoras. En estas condiciones, la seguridad no recibió mucha atención. Pero ahora, cuando millones de ciudadanos comunes usan redes para sus transacciones bancarias, compras y declaraciones de impuestos, la seguridad de las redes aparece en el horizonte como un problema potencial de grandes proporciones. En las siguientes secciones estudiaremos la seguridad de las redes desde varios ángulos, señalaremos muchos peligros y estudiaremos varios algoritmos y protocolos para hacer más seguras las redes.

La seguridad es un tema amplio que cubre una multitud de pecados. En su forma más sencilla, la seguridad se ocupa de garantizar que los curiosos no puedan leer, o peor aún, modificar mensajes dirigidos a otros destinatarios. Tiene que ver con la gente que intenta acceder a servicios remotos no autorizados. También se ocupa de mecanismos para verificar que el mensaje supuestamente enviado por la autoridad fiscal que indica: "Pague el viernes o aténgase a las consecuencias" realmente venga de ella y no de la mafia. La seguridad también se ocupa del problema de la captura y reproducción de mensajes legítimos, y de la gente que intenta negar el envío de mensajes.

La mayoría de los problemas de seguridad son causados intencionalmente por gente maliciosa que intenta ganar algo o hacerle daño a alguien. En la figura 8-1 se muestran algunos de los tipos de transgresores más comunes. Debe quedar claro por esta lista que hacer segura una red comprende mucho más que simplemente mantener los programas libres de errores de programación. Implica ser más listo que adversarios a menudo inteligentes, dedicados y a veces bien financiados. Debe quedar claro también que las medidas para detener a los adversarios casuales tendrán poca eficacia

contra los adversarios serios. Los registros policiales muestran que la mayoría de los ataques no son cometidos por intrusos que interfieren una línea telefónica sino por miembros internos con resentimientos. En consecuencia, los sistemas de seguridad deben diseñarse tomando en cuenta este hecho.

Adversario	Objetivo
Estudiante	Divertirse husmeando el correo de la gente
Cracker	Probar el sistema de seguridad de alguien; robar datos
Representante de ventas	Indicar que representa a toda Europa, no sólo a Andorra
Hombre de negocios	Descubrir el plan estratégico de marketing de un competidor
Ex empleado	Vengarse por haber sido despedido
Contador	Estafar dinero a una compañía
Corredor de bolsa	Negar una promesa hecha a un cliente por correo electrónico
Timador	Robar números de tarjeta de crédito
Espía	Conocer la fuerza militar o los secretos industriales de un enemigo
Terrorista	Robar secretos de guerra bacteriológica

Figura 8-1. Algunos tipos de personas que causan problemas de seguridad, y por qué.

Los problemas de seguridad de las redes pueden dividirse en términos generales en cuatro áreas interrelacionadas: confidencialidad, autenticación, no repudio y control de integridad. La confidencialidad consiste en mantener la información fuera de las manos de usuarios no autorizados. Esto es lo que normalmente viene a la mente cuando la gente piensa en la seguridad de las redes. La autenticación se encarga de determinar con quién se está hablando antes de revelar información delicada o hacer un trato de negocios. El no repudio se encarga de las firmas: ¿cómo comprobar que su cliente realmente hizo un pedido electrónico por 10 millones de utensilios para zurdos a 89 centavos cada uno cuando él luego aduce que el precio era de 69 centavos? O tal vez argumente que él nunca realizó ningún pedido. Por último, ¿cómo puede asegurarse de que un mensaje recibido realmente fue enviado, y no algo que un adversario malicioso modificó en el camino o cocinó por su propia cuenta?

Todos estos temas (confidencialidad, autenticación, no repudio y control de integridad) son pertinentes también en los sistemas tradicionales, pero con algunas diferencias importantes. La confidencialidad y la integridad se logran usando correo certificado y poniendo bajo llave los documentos. El robo del tren del correo es más difícil de lo que era en los tiempos de Jesse James.

También, la gente puede por lo general distinguir entre un documento original en papel y una fotocopia, y con frecuencia esto es importante. Como prueba, haga una fotocopia de un cheque válido. Trate de cobrar el cheque original en su banco el lunes. Ahora trate de cobrar la fotocopia del cheque el martes. Observe la diferencia de comportamiento del banco. Con los cheques electrónicos, el original y la copia son idénticos. Es posible que los bancos tarden un poco en acostumbrarse a esto.

La gente autentica la identidad de otras personas al reconocer sus caras, voces y letra. Las pruebas de firmas se manejan mediante firmas en papel, sellos, etc. Generalmente puede detectarse la alteración de documentos con el auxilio de expertos en escritura, papel y tinta. Ninguna de estas opciones está disponible electrónicamente. Es obvio que se requieren otras soluciones.

Antes de adentrarnos en las soluciones, vale la pena dedicar un instante a considerar el lugar que corresponde a la seguridad de las redes en la pila de protocolos. Probablemente no es un solo lugar. Cada capa tiene algo que contribuir. En la capa física podemos protegernos contra la intervención de las líneas de transmisión encerrando éstas en tubos sellados que contengan gas a alta presión. Cualquier intento de hacer un agujero en el tubo liberará un poco de gas, con lo cual la presión disminuirá y se disparará una alarma. Algunos sistemas militares usan esta técnica.

En la capa de enlace de datos, los paquetes de una línea punto a punto pueden encriptarse cuando se envíen desde una máquina y desencriptarse cuando lleguen a otra. Los detalles pueden manejarse en la capa de enlace de datos, sin necesidad de que las capas superiores se enteren de ello. Sin embargo, esta solución se viene abajo cuando los paquetes tienen que atravesar varios enrutadores, puesto que los paquetes tienen que desencriptarse en cada enrutador, dentro del cual son vulnerables a posibles ataques. Además, no se contempla que algunas sesiones estén protegidas (por ejemplo, aquellas que comprenden compras en línea mediante tarjeta de crédito) y otras no. No obstante, la **encriptación de enlace** (*link encryption*), como se llama a este método, puede agregarse fácilmente a cualquier red y con frecuencia es útil.

En la capa de red pueden instalarse *firewalls* para mantener adentro (o afuera) a los paquetes. La seguridad de IP también funciona en esta capa.

En la capa de transporte pueden encriptarse conexiones enteras, de extremo a extremo, es decir, proceso a proceso. Para lograr una máxima seguridad, se requiere seguridad de extremo a extremo.

Por último, los asuntos como la autenticación de usuario y el no repudio sólo pueden manejarse en la capa de aplicación.

Puesto que la seguridad no encaja por completo en ninguna capa, no podía integrarse en ningún capítulo de este libro. Por lo tanto, merece su propio capítulo.

Si bien este capítulo es largo, técnico y esencial, por el momento también es casi irrelevante. Por ejemplo, está bien documentado que la mayoría de las fallas de seguridad en los bancos se debe a empleados incompetentes, procedimientos de seguridad deficientes o a fraudes internos, más que a delincuentes inteligentes que intervienen líneas telefónicas y decodifican los mensajes encriptados. Si una persona entrara a un banco con una tarjeta de cajero automático que hubiera encontrado en la calle y solicitara un nuevo PIN argumentando haber olvidado el suyo, y éste se le proporcionara en el acto (en aras de las buenas relaciones con el cliente), toda la criptografía del mundo no podría evitar el abuso. A este respecto, el libro de Ross Anderson es una verdadera revelación, debido a que documenta cientos de ejemplos de fallas de seguridad en numerosas industrias, casi todas ellas debidas a lo que cortésmente podría llamarse prácticas de negocios descuidadas o falta de atención a la seguridad (Anderson, 2001). Sin embargo, pensamos con optimismo que a medida que el comercio electrónico se difunda con mayor amplitud, en algún momento las compañías depurarán sus procedimientos operativos, eliminando estos vacíos y dándole la importancia debida a los aspectos técnicos de seguridad.

Casi toda la seguridad se basa en principios de criptografía, a excepción de la seguridad en la capa física. Por esta razón, comenzaremos nuestro estudio de la seguridad examinando con detalle la criptografía. En la sección 8.1 veremos algunos de los principios básicos. En las secciones 8-2 a 8-5 examinaremos algunos de los algoritmos y estructuras de datos fundamentales que se utilizan en criptografía. A continuación examinaremos con detalle la forma en que pueden utilizarse estos conceptos para lograr la seguridad en la red. Finalizaremos con algunas ideas acerca de la tecnología y la sociedad.

Antes de comenzar, una última aclaración: qué es lo que no se cubre. Hemos tratado de enfocarnos en los aspectos de conectividad más que en los de sistema operativo o de aplicación, aunque la línea es muy sutil de distinguir. Por ejemplo, no hay nada aquí sobre la autenticación de usuario que utilice biométrica, seguridad mediante contraseña, ataques de sobreflujo de búfer, caballos de Troya, falsificación de inicio de sesión, bombas lógicas, virus, gusanos y cosas por el estilo. Todos estos temas se cubren con detalle en el capítulo 9 del libro *Sistemas Operativos Modernos* (Tanenbaum, 2003). El lector interesado en los aspectos de seguridad debe leer ese libro. Ahora comencemos nuestro viaje.

8.1 CRIPTOGRAFÍA

Criptografía viene del griego y significa "escritura secreta". Tiene una larga y colorida historia que se remonta a miles de años. En esta sección sólo delinearemos algunos de los puntos de interés, como antecedentes de lo que sigue. Si desea la historia completa de la criptografía, le recomiendo que lea el libro de Kahn (1995). Para un tratamiento más completo de los avances en lo que se refiere a los algoritmos de seguridad y criptografía, a los protocolos y a las aplicaciones, vea (Kaufman y cols., 2002). Para un enfoque más matemático, vea (Stinson, 2002). Para un enfoque menos matemático, vea (Burnett y Paine, 2001).

Los profesionales hacen una distinción entre cifrados y códigos. Un **cifrado** es una transformación carácter por carácter o bit por bit, sin importar la estructura lingüística del mensaje. En contraste, un **código** reemplaza una palabra con otra palabra o símbolo. Los códigos ya no se utilizan, aunque tienen una historia gloriosa. El código con mayor éxito fue el de las fuerzas armadas de Estados Unidos durante la Segunda Guerra Mundial en el Pacífico. Tenían indios navajos hablando entre sí, utilizando palabras específicas en navajo correspondientes a términos militares; por ejemplo, *chay-da-gahi-nail-tsaidi* (literalmente: asesino de tortugas) quiere decir arma antitanque. El lenguaje navajo es altamente tonal, extremadamente complejo y no tiene forma escrita. Ninguna persona en Japón sabía algo sobre él.

En septiembre de 1945, el *San Diego Union* describió el código diciendo "Durante tres años, en cualquier lugar donde los soldados de la Marina desembarcaban, los japoneses escuchaban una combinación de extraños gorjeos entremezclados con otros ruidos que se asemejaban al canto de un monje tibetano y al sonido de una botella de agua caliente al vaciarse". Los japoneses nunca descifraron el código y muchos de los soldados que utilizaban las claves navajo fueron premiados

con altos honores militares por su extraordinario servicio y valor. El hecho de que Estados Unidos descifrara el código japonés y que Japón no haya podido descifrar el navajo tuvo un papel importante en las victorias estadounidenses en el Pacífico.

8.1.1 Introducción a la criptografía

Históricamente, cuatros grupos de personas han utilizado y contribuido al arte de la criptografía: los militares, el cuerpo diplomático, los redactores de los periódicos y los amantes. De éstos, la milicia ha tenido el papel más importante y ha moldeado el campo a través de los siglos. Dentro de las organizaciones militares, los mensajes por encriptar se han entregado tradicionalmente a empleados mal pagados para su codificación y transmisión. El inmenso volumen de los mensajes ha impedido asignar esta labor a unos cuantos especialistas.

Hasta la llegada de las computadoras, una de las principales restricciones de la criptografía había sido la capacidad del empleado encargado de la codificación para realizar las transformaciones necesarias, con frecuencia en un campo de batalla con poco equipo. Una restricción adicional ha sido la dificultad de cambiar rápidamente de un método de criptografía a otro, debido a que esto implica volver a capacitar a una gran cantidad de personas. Sin embargo, el peligro de que un empleado fuera capturado por el enemigo ha hecho indispensable la capacidad de cambiar el método de criptografía de manera instantánea, de ser necesario. Estos requerimientos en conflicto han dado lugar al modelo de la figura 8-2.

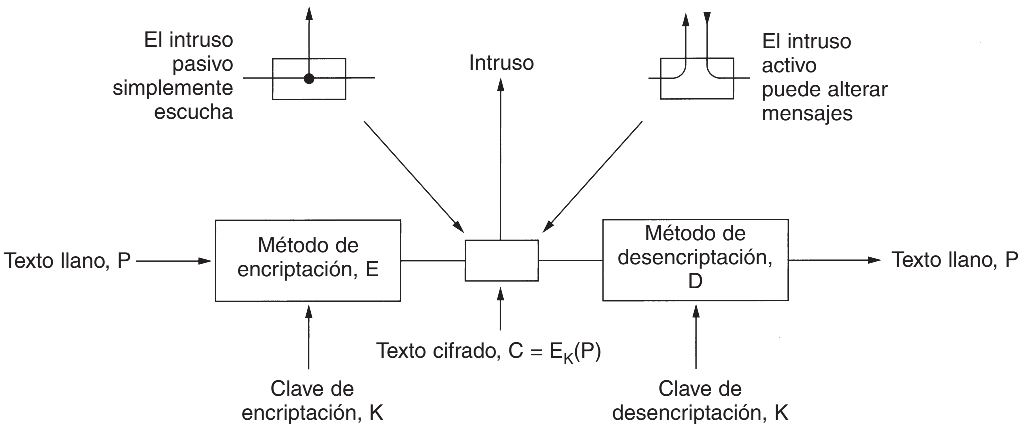


Figura 8-2. El modelo de encriptación (para un cifrado de clave simétrica).

Los mensajes por encriptar, conocidos como **texto llano,** son transformados por una función parametrizada por una **clave.** El resultado del proceso de encriptación, conocido como **texto cifrado,** se transmite a continuación, muchas veces por mensajero o radio. Suponemos que el enemigo, o **intruso,** escucha y copia con exactitud todo el texto cifrado. Sin embargo, a diferencia del destinatario original, el intruso no conoce la clave de desencriptación y no puede desencriptar con

facilidad el texto cifrado. En ocasiones el intruso no sólo escucha el canal de comunicación (intruso pasivo) sino también puede registrar mensajes y reproducirlos posteriormente, inyectar sus propios mensajes y modificar los mensajes legítimos antes de que lleguen al destinatario (intruso activo). El arte de descifrar los mensajes (**criptoanálisis**) y el arte de crear los cifrados (criptografía), se conocen en conjunto como **criptología.**

A menudo resulta útil tener una notación para relacionar el texto llano, el texto cifrado y las claves. Utilizaremos $C = E_K(P)$ para indicar que la encriptación del texto llano P usando la clave K produce el texto cifrado C. Del mismo modo, $P = D_K(C)$ representa la desencriptación de C para obtener el texto llano nuevamente. Por lo tanto,

$$D_K(E_K(P)) = P$$

Esta notación sugiere que E y D son sólo funciones matemáticas, lo cual es cierto. El único truco es que ambas son funciones de dos parámetros, y hemos escrito uno de los parámetros (la clave) como subíndice, en lugar de como argumento, para distinguirlo del mensaje.

Una regla fundamental de la criptografía es que se debe suponer que el criptoanalista conoce el método general de encriptación y desencriptación usado. En otras palabras, el criptoanalista sabe con detalle cómo funciona el método de encriptación, E, y desencriptación, D, de la figura 8-2. La cantidad de esfuerzo necesario para inventar, probar e instalar un nuevo algoritmo cada vez que el método antiguo está en peligro, o se piensa que lo está, siempre ha hecho impráctico mantener en secreto el algoritmo de encriptación. Además, pensar que dicho algoritmo es secreto cuando no lo es, hace más daño que bien.

Aquí es donde entra la clave. Ésta consiste en una cadena corta (relativamente) que selecciona una de muchas encriptaciones potenciales. En contraste con el método general, que tal vez se cambie cada cierto número de años, la clave puede cambiarse con la frecuencia requerida. Por lo tanto, nuestro modelo básico es un método general estable y conocido públicamente pero parametrizado por una clave secreta y que puede cambiarse con facilidad. La idea de que el criptoanalista conozca los algoritmos y que la naturaleza secreta se base principalmente en las claves se conoce como **principio de Kerckhoff,** que debe su nombre al criptógrafo militar holandés Auguste Kerckhoff, que lo estableció en 1883 (Kerckhoff, 1883). Por lo tanto, tenemos:

> *Principio de Kerckhoff: Todos los algoritmos deben ser públicos; sólo las claves deben ser secretas*

La naturaleza no secreta del algoritmo no puede remarcarse lo suficiente. Tratar de mantener secreto el algoritmo, lo que se conoce como **seguridad por desconocimiento,** nunca funciona. Además, al hacer público el algoritmo, el criptógrafo recibe asesoría gratuita de una gran cantidad de criptólogos académicos ansiosos por descifrar el sistema con el propósito de publicar trabajos que demuestren su inteligencia. Si muchos expertos han tratado de descifrar el algoritmo durante cinco años después de su publicación y nadie lo ha logrado, probablemente es bastante sólido.

Puesto que la parte secreta debe ser la clave, la longitud de ésta es un aspecto importante del diseño. Considere una cerradura de combinación. El principio general es que se introducen dígitos

en secuencia. Todo el mundo lo sabe, pero la clave es secreta. Una longitud de clave de dos dígitos significa que hay 100 posibilidades. Una clave de tres dígitos significa que hay 1000 posibilidades y una clave de seis dígitos de longitud significa un millón. Cuanto más grande sea la clave, mayor será el **factor de trabajo** que tendrá que enfrentar el criptoanalista. El factor de trabajo para descifrar el sistema mediante una búsqueda exhaustiva del espacio de clave crece exponencialmente con la longitud de la clave. El secreto radica en tener un algoritmo robusto (pero público) y una clave larga. Para evitar que su hermano menor lea su correo electrónico, las claves de 64 bits son suficientes. Para el uso comercial común, se requieren por lo menos 128 bits. Para mantener a raya a gobiernos poderosos se requieren claves de al menos 256 bits, y de preferencia mayores.

Desde el punto de vista del criptoanalista, el problema del criptoanálisis presenta tres variaciones principales. Cuando el criptoanalista cuenta con cierta cantidad de texto cifrado pero no tiene texto llano, enfrenta el problema de **sólo texto cifrado.** Los criptogramas que aparecen en la sección de acertijos de los diarios presentan este tipo de problema. Cuando el criptoanalista cuenta con texto cifrado y el texto llano correspondiente, se enfrenta al problema del **texto llano conocido.** Por último, cuando el criptoanalista tiene la capacidad de encriptar textos normales que él escoge, enfrenta el problema del **texto llano seleccionado.** Los criptogramas de los periódicos podrían descifrarse con facilidad si el criptoanalista pudiera hacer preguntas como: ¿cuál es la encriptación de ABCDEFGHIJKL?

Los principiantes en el campo de la criptografía con frecuencia suponen que, si un cifrado puede resistir el ataque de sólo texto cifrado, entonces es seguro. Esta suposición es muy ingenua. En muchos casos, el criptoanalista puede hacer una buena estimación de partes de texto llano. Por ejemplo, lo primero que muchas computadoras dicen cuando se les llama es "Indique su clave:". Equipado con algunos pares concordantes de texto llano-texto cifrado, el trabajo del criptoanalista se vuelve mucho más fácil. Para lograr seguridad, el criptógrafo debe ser conservador y asegurarse de que el sistema sea inviolable aun si su oponente puede encriptar cantidades arbitrarias de texto llano seleccionado.

Los métodos de encriptación han sido divididos históricamente en dos categorías: cifrados por sustitución y cifrados por transposición. A continuación reseñaremos cada uno de éstos como antecedente para la criptografía moderna.

8.1.2 Cifrados por sustitución

En un **cifrado por sustitución,** cada letra o grupo de letras se reemplazan por otra letra o grupo de letras para disfrazarla. Uno de los cifrados más viejos conocidos es el **cifrado de César,** atribuido a Julio César. En este método, *a* se vuelve *D*, *b* se vuelve *E*, *c* se vuelve *F*, ... , y *z* se vuelve *C*. Por ejemplo, *ataque* se vuelve *DWDTXH*. En los ejemplos, el texto llano se presentará en minúsculas y el texto cifrado en mayúsculas.

Una pequeña generalización del cifrado de César permite que el alfabeto de texto cifrado se desplace *k* letras, en lugar de siempre 3. En este caso, *k* se convierte en una clave del método general de alfabetos desplazados circularmente. El cifrado de César posiblemente engañó a Pompeyo, pero no ha engañado a nadie desde entonces.

La siguiente mejora es hacer que cada uno de los símbolos del texto llano, digamos las 26 letras del abecedario (no incluimos la *ñ*) inglés, tengan una correspondencia con alguna otra letra. Por ejemplo,

texto llano:	a b c d e f g h i j k l m n o p q r s t u v w x y z
texto cifrado:	Q W E R T Y U I O P A S D F G H J K L Z X C V B N M

El sistema general de sustitución de símbolo por símbolo se llama **sustitución monoalfabética,** siendo la clave la cadena de 26 letras correspondiente al alfabeto completo. Para la clave anterior, el texto llano *ataque* se transformaría en el texto cifrado *QZQJXT.*

A primera vista, esto podría parecer un sistema seguro, porque, aunque el criptoanalista conoce el sistema general (sustitución letra por letra), no sabe cuál de las $26! \approx 4 \times 10^{26}$ claves posibles se está usando. En contraste con el cifrado de César, intentarlas todas no es una estrategia muy alentadora. Aun a 1 nseg por solución, una computadora tardaría 10^{10} años en probar todas las claves.

No obstante, si se cuenta con una cantidad aun pequeña de texto cifrado, puede descifrarse fácilmente. El ataque básico aprovecha las propiedades estadísticas de los lenguajes naturales. En inglés, por ejemplo, la *e* es la letra más común, seguida de *t, o, a, n, i,* etc. Las combinaciones de dos letras más comunes, o **digramas,** son *th, in, er, re* y *an.* Las combinaciones de tres letras más comunes, o **trigramas,** son *the, ing, and* e *ion.*

Un criptoanalista que intenta descifrar un cifrado monoalfabético comenzaría por contar la frecuencia relativa de todas las letras del texto cifrado. Entonces podría asignar tentativamente la más común a la letra *e* y la siguiente más común a la letra *t*. Vería entonces los trigramas para encontrar uno común de la forma *tXe,* lo que sugerirá fuertemente que *X* es *h*. De la misma manera, si el patrón *thYt* ocurre con frecuencia, *Y* probablemente representa *a*. Con esta información se puede buscar un trigrama frecuente de la forma *aZW*, que con mucha probabilidad es *and*. Adivinando las palabras comunes, digramas y trigramas, y conociendo los patrones probables de las vocales y consonantes, el criptoanalista construye un texto llano tentativo, letra por letra.

Otra estrategia es adivinar una palabra o frase probable. Por ejemplo, considere el siguiente texto cifrado de una compañía contable (en bloques de cinco caracteres):

```
CTBMN BYCTC BTJDS QXBNS GSTJC BTSWX CTQTZ CQVUJ
QJSGS TJQZZ MNQJS VLNSX VSZJU JDSTS JQUUS JUBXJ
DSKSU JSNTK BGAQJ ZBGYQ TLCTZ BNYBN QJSW
```

Una palabra muy probable en un mensaje de una compañía contable de un país de habla inglesa es *financial*. Usando nuestro conocimiento de que *financial* tiene una letra repetida (*i*), con cuatro letras intermedias entre su aparición, buscamos letras repetidas en el texto cifrado con este espaciado. Encontramos 12 casos, en las posiciones 6, 15, 27, 31, 42, 48, 56, 66, 70, 71, 76 y 82. Sin embargo, sólo dos de éstos, el 31 y 42, tienen la siguiente letra (correspondiente a *n* en el texto llano) repetida en el lugar adecuado. De estos dos, sólo el 31 tiene también la *a* en la posición

correcta, por lo que sabemos que *financial* comienza en la posición 30. A partir de aquí, la deducción de la clave es fácil usando las estadísticas de frecuencia del texto en inglés.

8.1.3 Cifrados por transposición

Los cifrados por sustitución conservan el orden de los símbolos de texto llano, pero los disfrazan. Los **cifrados por transposición,** en contraste, reordenan las letras pero no las disfrazan. En la figura 8-3 se presenta un cifrado de transposición común, la transposición columnar. La clave del cifrado es una palabra o frase que no contiene letras repetidas. En este ejemplo, la clave es MEGABUCK. El propósito de la clave es numerar las columnas, estando la columna 1 bajo la letra clave más cercana al inicio del alfabeto, y así sucesivamente. El texto llano se escribe horizontalmente, en filas, las cuales se rellenan para completar la matriz si es necesario. El texto cifrado se lee por columnas, comenzando por la columna cuya letra clave es la más baja.

M	E	G	A	B	U	C	K
7	4	5	1	2	8	3	6
p	l	e	a	s	e	t	r
a	n	s	f	e	r	o	n
e	m	i	l	l	i	o	n
d	o	l	l	a	r	s	t
o	m	y	s	w	i	s	s
b	a	n	k	a	c	c	o
u	n	t	s	i	x	t	w
o	t	w	o	a	b	c	d

Texto llano

pleasetransferonemilliondollarsto
myswissbankaccountsixtwotwo

Texto cifrado

AFLLSKSOSELAWAIATOOSSCTCLNMOMANT
ESILYNTWRNNTSOWDPAEDOBUOERIRICXB

Figura 8-3. Cifrado por transposición.

Para descifrar un cifrado por transposición, el criptoanalista debe primero estar consciente de que está tratando con un cifrado de este tipo. Observando la frecuencia de *E, T, A, O, I, N,* etc., es fácil ver si se ajustan al patrón usual del texto llano. De ser así, es evidente que se trata de un cifrado por transposición, pues en tal cifrado cada letra se representa a sí misma y la distribución de frecuencia permanece intacta.

El siguiente paso es adivinar la cantidad de columnas. En muchos casos, puede adivinarse una palabra o frase probable por el contexto del mensaje. Por ejemplo, supóngase que nuestro criptoanalista sospecha que la frase de texto llano *milliondollars* aparece en algún lugar del mensaje. Observe que los digramas *MO, IL, LL, LA, IR* y *OS* aparecen en el texto cifrado como resultado de la envoltura de la frase. La letra de texto cifrado *O* sigue a la letra de texto cifrado *M* (es decir, son

adyacentes verticalmente en la columna 4) puesto que están separados en la frase probable por una distancia igual a la longitud de la clave. Si se hubiera usado una clave de longitud siete, habrían aparecido los digramas *MD, IO, LL, LL, IA, OR* y *NS*. De hecho, para cada longitud de clave, se produce un grupo diferente de digramas en el texto cifrado. Buscando las diferentes posibilidades, el criptoanalista con frecuencia puede determinar fácilmente la longitud de la clave.

El paso restante es ordenar las columnas. Cuando la cantidad de columnas, k, es pequeña, puede examinarse cada uno de los pares de columnas $k(k-1)$ para ver si la frecuencia de sus digramas es igual a la del texto llano. El par con la mejor concordancia se supone correctamente ubicado. Ahora cada columna restante se prueba tentativamente como sucesora de este par. La columna cuyas frecuencias de digramas y trigramas produce la mejor concordancia se toma tentativamente como correcta. La columna antecesora se encuentra de la misma manera. El proceso completo se repite hasta encontrar un orden potencial. Es probable que en este punto se pueda reconocer texto llano (por ejemplo, si aparece *milloin*, quedará claro en dónde está el error).

Algunos cifrados por transposición aceptan un bloque de longitud fija como entrada y producen un bloque de longitud fija como salida. Estos cifrados pueden describirse por completo con sólo dar una lista que indique el orden en el que deben salir los caracteres. Por ejemplo, el cifrado de la figura 8-3 puede verse como un cifrado de bloque de 64 caracteres. Su salida es 4, 12, 20, 28, 36, 44, 52, 60, 5, 13, . . . , 62. En otras palabras, el cuarto carácter de entrada, *a,* es el primero en salir, seguido del decimosegundo, *f*, etcétera.

8.1.4 Rellenos de una sola vez

La construcción de un cifrado inviolable en realidad es bastante sencilla; la técnica se conoce desde hace décadas. Primero se escoge una cadena de bits al azar como clave. Luego se convierte el texto llano en una cadena de bits, por ejemplo usando su representación ASCII. Por último, se calcula el OR EXCLUSIVO de estas dos cadenas, bit por bit. El texto cifrado resultante no puede descifrarse, porque en una muestra suficientemente grande de texto cifrado cada letra aparecerá con la misma frecuencia, lo mismo que cada digrama y trigrama, y así sucesivamente. Este método, conocido como **relleno de una sola vez,** es inmune a todos los ataques actuales y futuros sin importar cuánta potencia computacional tenga el intruso. La razón se deriva de una teoría de la información: simplemente no hay información en el mensaje debido a que todos los textos llanos posibles de una longitud dada son parecidos.

En la figura 8-4 se muestra un ejemplo de cómo se utilizan los rellenos de una sola vez. Primero, el mensaje 1, "I love you." se convierte a ASCII de 7 bits. A continuación se elige el relleno de una sola vez, relleno 1, y se efectúa un XOR con el mensaje para obtener el texto cifrado. Un criptoanalista podría probar todos los rellenos de una sola vez posibles para ver qué texto llano proviene de cada uno. Por ejemplo, podría probarse el relleno 2 de la figura, lo que resultaría en el texto llano 2, "Elvis lives", lo cual podría ser o no verosímil (un tema que está fuera del alcance de este libro). De hecho, para cada texto llano ASCII de 11 caracteres hay un relleno de una sola vez que lo genera. Eso es lo que queremos dar a entender cuando decimos que no hay

información en el texto cifrado: es posible obtener cualquier mensaje con la longitud correcta a partir de él.

Mensaje 1:	1001001 0100000 1101100 1101111 1110110 1100101 0100000 1111001 1101111 1110101 0101110
Relleno 1:	1010010 1001011 1110010 1010101 1010010 1100011 0001011 0101010 1010111 1100110 0101011
Texto cifrado:	0011011 1101011 0011110 0111010 0100100 0000110 0101011 1010011 0111000 0010011 0000101
Relleno 2:	1011110 0000111 1101000 1010011 1010111 0100110 1000111 0111010 1001110 1110110 1110110
Texto llano 2:	1000101 1101100 1110110 1101001 1110011 0100000 1101100 1101001 1110110 1100101 1110011

Figura 8-4. El uso de un relleno de una sola vez para cifrado y la posibilidad de obtener cualquier texto llano posible a partir del texto cifrado mediante el uso de un relleno diferente.

En teoría, los rellenos de una sola vez son excelentes, pero en la práctica tienen varias desventajas. Para comenzar, la clave no se puede memorizar, por lo que tanto el emisor como el receptor deben cargar una copia escrita con ellos. Si cualquiera de ellos corre el peligro de ser capturado, es obvio que las claves escritas no son una buena idea. Además, la cantidad total de datos que se pueden transmitir está limitada por la cantidad de claves disponibles. Si el espía tiene suerte y descubre una gran cantidad de datos, quizá no los pueda transmitir al cuartel general debido a que la clave se ha agotado. Otro problema es la sensibilidad del método a los caracteres perdidos o insertados. Si el emisor y el receptor pierden la sincronización, de ahí en adelante todos los datos aparecerán distorsionados.

Con la aparición de las computadoras, el relleno de una sola vez podría volverse práctico para algunas aplicaciones. El origen de la clave podría ser un DVD especial que contenga varios gigabytes de información y si se transporta en una caja de película para DVD y se antecede con algunos minutos de vídeo, no sería sospechoso. Por supuesto, a velocidades de red de gigabit, tener que insertar un nuevo DVD cada 30 seg podría volverse tedioso. Y los DVDs deben transportarse personalmente desde el emisor hasta el receptor antes de que cualquier mensaje pueda ser enviado, lo que reduce en gran medida su utilidad práctica

Criptografía cuántica

Podría haber una solución al problema de la transmisión del relleno de una sola vez a través de la red, y proviene de un origen muy inverosímil: la mecánica cuántica. Esta área aún es experimental, pero las pruebas iniciales son alentadoras. Si pudiera perfeccionarse y fuera eficiente, con el tiempo casi toda la criptografía se realizará utilizando rellenos de una sola vez debido a su seguridad. A continuación explicaremos brevemente cómo funciona el método de la **criptografía cuántica.** En particular, describiremos un protocolo llamado **BB84** que debe su nombre a sus autores y a su año de publicación (Bennet y Brassard, 1984).

Un usuario, Alice, desea establecer un relleno de una sola vez con un segundo usuario, Bob. Alice y Bob son los **personajes principales.** Por ejemplo, Bob es un banquero con quien Alice quiere hacer negocios. En la década pasada, los nombres "Alice" y "Bob" se utilizaron como

personajes principales en casi todo lo escrito sobre criptografía. Los criptógrafos aman la tradición. Si utilizáramos "Andy" y "Barbara" como personajes principales, nadie creería nada de lo que se dice en este capítulo. Por lo tanto, dejémoslo así.

Si Alice y Bob pudieran establecer un relleno de una sola vez, podrían utilizarlo para comunicarse de manera segura. La pregunta es: ¿Cómo pueden establecerlo sin intercambiar antes DVDs? Podemos asumir que Alice y Bob se encuentran en extremos opuestos de un cable de fibra óptica a través del cual pueden enviar y recibir pulsos de luz. Sin embargo, un intruso intrépido, Trudy, puede cortar la fibra para establecer una derivación activa. Trudy puede leer todos los bits en ambas direcciones. También puede enviar mensajes falsos en ambas direcciones. Para Alice y Bob la situación podría parecer irremediable, pero la criptografía cuántica puede arrojarle un poco de luz.

La criptografía cuántica se basa en el hecho de que la luz viene en pequeños paquetes llamados **fotones,** los cuales tienen algunas propiedades peculiares. Además, la luz puede polarizarse al pasarla a través de un filtro de polarización, un hecho bien conocido por quienes utilizan anteojos oscuros y por los fotógrafos. Si se pasa un haz de luz (es decir, un flujo de fotones) a través de un filtro polarizador, todos los fotones que emerjan de él se polarizarán en la dirección del eje del filtro (por ejemplo, el vertical). Si a continuación el haz se pasa a través de un segundo filtro polarizador, la intensidad de la luz que emerja del segundo filtro es proporcional al cuadrado del coseno del ángulo entre los ejes. Si los dos ejes son perpendiculares, no pasa ningún fotón. La orientación absoluta de los dos filtros no importa; sólo cuenta el ángulo entre sus ejes.

Para generar un relleno de una sola vez, Alice necesita dos conjuntos de filtros polarizados. El primer conjunto consiste en un filtro vertical y en uno horizontal. Esta opción se conoce como **base rectilínea.** Una base es simplemente un sistema de coordenadas. El segundo conjunto de filtros consiste en lo mismo, excepto que se gira 45 grados, de forma que un filtro va del extremo inferior izquierdo al extremo superior derecho y el otro va del extremo superior izquierdo al extremo inferior derecho. Esta opción se conoce como **base diagonal.** Por lo tanto, Alice tiene dos bases, las cuales puede insertar a voluntad y rápidamente en su haz. En la realidad, Alice no tiene cuatro filtros separados, sino un cristal cuya polarización puede cambiarse de manera eléctrica y a gran velocidad a cualquiera de las cuatro direcciones permitidas. Bob tiene el mismo equipo que Alice. El hecho de que Alice y Bob dispongan de dos bases cada uno es esencial para la criptografía cuántica.

Ahora Alice asigna una dirección como 0 y la otra como 1 para cada base. En el ejemplo siguiente suponemos que elige 0 para vertical y 1 para horizontal. De manera independiente, también elige 0 para la dirección del extremo inferior izquierdo al superior derecho y 1 para la dirección del extremo superior izquierdo al inferior derecho. Alice envía estas opciones a Bob como texto llano.

Ahora Alice elige un relleno de una sola vez, por ejemplo, con base en un generador de números aleatorios (un tema complejo por sí mismo). Lo transmite bit por bit a Bob, eligiendo de manera aleatoria una de sus dos bases para cada bit. Para enviar un bit, su pistola de fotones emite un fotón polarizado apropiado para la base que está utilizando para ese bit. Por ejemplo, podría elegir bases diagonal, rectilínea, rectilínea, diagonal, rectilínea, etcétera. Para enviarle un relleno de una sola vez de 1001110010100110 con estas bases, Alice debería enviar los fotones que se muestran en la figura 8-5(a). Dados el relleno de una sola vez y la secuencia de bases, la polarización

por utilizar para cada bit se determina de manera única. Los bits enviados un fotón a la vez se conocen como **qubits.**

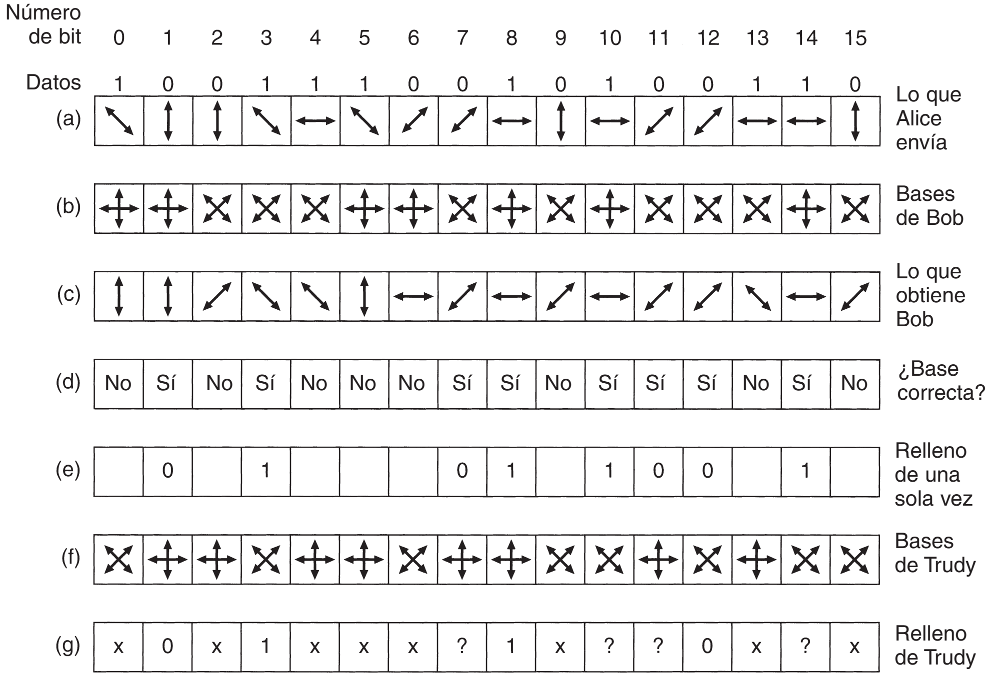


Figura 8-5. Ejemplo de criptografía cuántica.

Bob no sabe cuáles bases utilizar, por lo que elige una al azar para cada fotón entrante y simplemente la usa, como se muestra en la figura 8-5(b). Si elige la base correcta, obtiene el bit correcto. Si elige la base incorrecta, obtiene un bit aleatorio debido a que si un fotón choca con un filtro que está polarizado a 45 grados de su propia polarización, salta de manera aleatoria a la polarización del filtro o a una polarización perpendicular al filtro con igual probabilidad. Esta propiedad de los fotones es fundamental para la mecánica cuántica. Por lo tanto, algunos de los bits son correctos y algunos son aleatorios, pero Bob no sabe cuál es cuál. Los resultados de Bob se muestran en la figura 8-5(c).

¿Cómo sabe Bob cuáles de las bases que obtuvo son correctas y cuáles erróneas? Simplemente indica a Alice, mediante texto llano, cuál base utilizó para cada bit y ella le indica a él, también en texto llano, cuáles son correctas y cuáles erróneas, como se muestra en la figura 8-5(d). A partir de esta información los dos pueden construir una cadena de bits tomando en cuenta los aciertos, como se muestra en la figura 8-5(e). En promedio, esta cadena de bits tendrá la mitad del tamaño de la cadena de bits original, pero debido a que las dos partes lo saben, ambas pueden utilizarla como un relleno de una sola vez. Todo lo que Alice tiene que hacer es transmitir una cadena de bits un poco mayor que el doble de la longitud deseada y ella y Bob tendrán un relleno de una sola vez con la longitud deseada. Problema resuelto.

Pero espere un momento. Nos olvidamos de Trudy. Suponga que ella tiene curiosidad por saber lo que dirá Alice y corta la fibra, insertando sus propios detector y transmisor. Desgraciadamente para ella, tampoco sabe cuál base utilizar para cada fotón. Lo mejor que puede hacer es elegir una al azar para cada fotón, tal como lo hace Bob. En la figura 8-5(f) se muestra un ejemplo de sus elecciones. Cuando más tarde Bob informa (en texto llano) cuáles bases utilizó y Alice le indica (en texto llano) cuáles son correctas, Trudy sabe cuándo acertó y cuándo no. En la figura 8-5 acertó en los bits 0, 1, 2, 3, 4, 6, 8, 12 y 13. Pero sabe a partir de la respuesta de Alice en la figura 8-5(d) que sólo los bits 1, 3, 7, 8, 10, 11, 12 y 14 son parte del relleno de una sola vez. Acertó en cuatro de estos bits (1, 3, 8 y 12). En los otros cuatro (7, 10, 11 y 14) no coincidió y no conoce el bit transmitido. Por lo tanto, a partir de la figura 8-5(e), Bob sabe que el relleno de una sola vez inicia con 01011001, pero todo lo que Trudy tiene es 01?1??0?, tomando en cuenta la figura 8-5(g).

Por supuesto, Alice y Bob están conscientes de que tal vez Trudy haya capturado parte de su relleno de una sola vez, por lo que les gustaría reducir la información que Trudy tiene. Esto lo pueden conseguir realizando una transformación sobre el relleno. Por ejemplo, pueden dividir el relleno de una sola vez en bloques de 1024 bits y elevar al cuadrado cada uno de ellos para formar un número de 2048 bits y utilizar la concatenación de estos números de 2048 bits como el relleno de una sola vez. Con su conocimiento parcial de la cadena de bits transmitida, Trudy no tiene forma de elevar al cuadrado y, por lo tanto, no tiene nada. La transformación del relleno original de una sola vez a uno diferente que reduce el conocimiento de Trudy se conoce como **amplificación de privacidad.** En la práctica, en lugar de las elevaciones al cuadrado se utilizan transformaciones complejas en las que cada bit de salida depende de cada bit de entrada.

Pobre Trudy. No sólo no tiene idea de cuál es el relleno de una sola vez, sino que su presencia tampoco es un secreto. Después de todo, debe transmitir cada bit recibido a Bob para hacerle creer que está hablando con Alice. El problema es que lo mejor que puede hacer es transmitir el qubit que recibió, utilizando la polarización que utilizó para recibirlo, y casi la mitad del tiempo estará mal, causando muchos errores en el relleno de una sola vez de Bob.

Cuando Alice finalmente comienza a enviar datos, los codifica mediante un pesado código de corrección de errores hacia adelante. Desde el punto de vista de Bob, un error de un 1 bit en el relleno de una sola vez es lo mismo que un error de transmisión de 1 bit. De cualquier forma, obtiene el bit incorrecto. Si hay suficiente corrección de errores hacia adelante, puede recuperar el mensaje original a pesar de todos los errores, y puede contar fácilmente cuántos errores fueron corregidos. Si este número es mayor que la tasa de errores esperada del equipo, él sabe que Trudy ha intervenido la línea y puede actuar en consecuencia (por ejemplo, decirle a Alice que cambie a un canal de radio, que llame a la policía, etcétera). Si Trudy tuviera alguna forma de clonar un fotón a fin de contar con un fotón para inspeccionar y uno idéntico para enviar a Bob, podría evitar la detección, pero en la actualidad no se conoce ninguna forma de clonar perfectamente un fotón. Sin embargo, aunque Trudy pudiera clonar fotones, no se podría reducir el valor de la criptografía cuántica para establecer rellenos de una sola vez.

Aunque se ha mostrado que la criptografía cuántica puede operar a través de distancias de 60 km de fibra, el equipo es complejo y costoso. Aun así, la idea es prometedora. Para obtener mayor información sobre la criptografía cuántica, vea (Mullins, 2002).

8.1.5 Dos principios criptográficos fundamentales

Aunque estudiaremos muchos sistemas criptográficos diferentes en las siguientes páginas, hay dos principios que los sostienen a todos y que es importante entender.

Redundancia

El primer principio es que todos los mensajes encriptados deben contener redundancia, es decir, información no necesaria para entender el mensaje. Un ejemplo puede dejar en claro la necesidad de esto. Considere una compañía de compras por correo, El Perezoso (EP), que tiene 60,000 productos. Pensando que son muy eficientes, los programadores de EP deciden que los mensajes de pedidos deben consistir en un nombre de cliente de 16 bytes seguido de un campo de datos de 3 bytes (1 byte para la cantidad y 2 para el número de producto). Los últimos 3 bytes deben encriptarse usando una clave muy grande conocida sólo por el cliente y EP.

En un principio, esto podría parecer seguro, y en cierto sentido lo es, puesto que los intrusos pasivos no pueden descifrar los mensajes. Desgraciadamente, el sistema también tiene una falla mortal que lo vuelve inútil. Supongamos que un empleado recientemente despedido quiere vengarse de EP por haberlo cesado. Antes de irse, se lleva con él (parte de) la lista de clientes; entonces trabaja toda la noche escribiendo un programa para generar pedidos ficticios usando nombres reales de los clientes. Dado que no tiene la lista de claves, simplemente pone números aleatorios en los últimos 3 bytes y envía cientos de pedidos a EP.

Al llegar estos mensajes, la computadora de EP usa el nombre del cliente para localizar la clave y descifrar el mensaje. Para mala suerte de EP, casi cada mensaje de 3 bytes es válido, por lo que la computadora comienza a imprimir instrucciones de embarque. Aunque podría parecer extraño que un cliente ordene 837 juegos de columpios para niños, o 540 cajas de arena, en lo que a la computadora concierne el cliente bien podría estar planeando abrir una cadena de franquicias con juegos infantiles. De esta manera, un intruso activo (el ex empleado) puede causar muchísimos problemas, aun cuando no pueda entender los mensajes que genera su computadora.

Este problema puede resolverse agregando redundancia a todos los mensajes. Por ejemplo, si se extienden los mensajes de pedido a 12 bytes, de los cuales los primeros 9 deben ser ceros, entonces este ataque ya no funciona, porque el ex empleado ya no puede generar una cadena grande de mensajes válidos. La moraleja de esta historia es que todos los mensajes deben contener una cantidad considerable de redundancia para que los intrusos activos no puedan enviar basura al azar y lograr que se interprete como mensajes válidos.

Sin embargo, la adición de redundancia también simplifica a los criptoanalistas el descifrado de los mensajes. Supongamos que el negocio de pedidos por correo es altamente competido, y que la competencia principal de El Perezoso, El Bolsón, estaría encantado de conocer la cantidad de cajas de arena que EP vende. Para ello, ha intervenido la línea telefónica de EP. En el esquema original con mensajes de 3 bytes, el criptoanálisis era prácticamente imposible puesto que, tras adivinar una clave, el criptoanalista no tenía manera de saber si había adivinado correctamente. A fin de cuentas, casi cualquier mensaje era técnicamente legal. Con el nuevo esquema de 12 bytes,

es fácil para el criptoanalista distinguir un mensaje válido de uno no válido. Por lo tanto, tenemos:

Principio criptográfico 1: Los mensajes deben contener alguna redundancia

En otras palabras, al desencriptar un mensaje, el destinatario debe tener la capacidad de saber si es válido con sólo inspeccionarlo y tal vez realizando un cálculo simple. Esta redundancia es necesaria para evitar que intrusos activos envíen basura y engañen al receptor para que descifre la basura y realice algo con el "texto llano". Sin embargo, esta misma redundancia simplifica en gran medida la violación del sistema por parte de los intrusos pasivos, por lo que aquí hay un poco de preocupación. Más aún, la redundancia nunca debe estar en la forma de *n* ceros al principio o al final del mensaje, debido a que al ejecutar tales mensajes a través de algunos algoritmos criptográficos los resultados son más predecibles, lo que facilita el trabajo del criptoanalista. Un polinomio CRC es mejor que una serie de 0s puesto que el receptor puede verificarlo fácilmente, pero implica más trabajo para el criptoanalista. Un método aún mejor es utilizar un *hash* criptográfico, concepto que exploraremos más adelante.

Regresando a la criptografía cuántica por un momento, también podemos ver que la redundancia juega un papel ahí. Debido a que Trudy interceptó los fotones, algunos bits en el relleno de una sola vez de Bob serán incorrectos. Bob necesita algo de redundancia en los mensajes entrantes para determinar que se presentaron errores. Una forma muy burda de redundancia es repetir dos veces el mensaje. Si ambas copias no son idénticas, Bob sabe que o la fibra tiene mucho ruido o que alguien está interviniendo la transmisión. Por supuesto, enviar todo dos veces es excesivo; los códigos de Hamming o de Reed-Solomon constituyen formas más eficientes para realizar la detección y corrección de errores. Pero debe quedar claro que para distinguir un mensaje válido de uno inválido, especialmente en el caso de que haya un intruso activo, es necesaria cierta cantidad de redundancia.

Actualización

El segundo principio criptográfico es que se deben tomar medidas para asegurar que cada mensaje recibido se verifique a fin de saber si está actualizado, es decir, que se haya enviado muy recientemente. Esta medida es necesaria para evitar que intrusos activos reproduzcan mensajes antiguos. Si no se tomaran tales medidas, el ex empleado podría intervenir la línea telefónica de EP y simplemente continuar repitiendo los mensajes válidos enviados previamente. En otras palabras, tenemos:

Principio criptográfico 2: Es necesario algún método para frustrar los ataques de repetición

Una de tales medidas es incluir en cada mensaje una marca de tiempo válida durante, digamos, 10 segundos. El receptor puede entonces guardar los mensajes unos 10 segundos, para compararlos con los mensajes nuevos que lleguen y filtrar los duplicados. Los mensajes con una antigüedad mayor a 10 segundos pueden descartarse, dado que todas las repeticiones enviadas más de 10 segundos después también se rechazarán como demasiado viejas. Más adelante estudiaremos algunas medidas diferentes a las marcas de tiempo.

8.2 ALGORITMOS DE CLAVE SIMÉTRICA

La criptografía moderna usa las mismas ideas básicas que la criptografía tradicional (la transposición y la sustitución), pero su orientación es distinta. Tradicionalmente, los criptógrafos han usado algoritmos. Hoy día se hace lo opuesto: el objetivo es hacer el algoritmo de encriptación tan complicado y rebuscado que incluso si el criptoanalista obtiene cantidades enormes de texto cifrado a su gusto, no será capaz de entender nada sin la clave.

La primera clase de algoritmos de encriptación que analizaremos en este capítulo se conocen como **algoritmos de clave simétrica** porque utilizan la misma clave para encriptar y desencriptar. La figura 8-2 ilustra el uso de un algoritmo de clave simétrica. En particular, nos enfocaremos en los **cifrados en bloques,** que toman un bloque de *n* bits de texto llano como entrada y lo transforman utilizando la clave en un bloque de *n* bits de texto cifrado.

Los algoritmos criptográficos pueden implementarse ya sea en hardware (para velocidad) o en software (para flexibilidad). Aunque la mayoría de nuestro tratamiento tiene que ver con algoritmos y protocolos, que son independientes de la implementación real, no están de más unas cuantas palabras acerca de la construcción de hardware criptográfico. Las transposiciones y las sustituciones pueden implementarse mediante circuitos eléctricos sencillos. En la figura 8-6(a) se muestra un dispositivo, conocido como **caja P** (la P significa permutación), que se utiliza para efectuar una transposición de una entrada de 8 bits. Si los 8 bits se designan de arriba hacia abajo como 01234567, la salida de esta caja P en particular es 36071245. Mediante el cableado interno adecuado, puede hacerse que una caja P efectúe cualquier transposición prácticamente a la velocidad de la luz, pues no se requiere ningún cálculo, sólo propagación de la señal. Este diseño sigue el principio de Kerckhoff: el atacante sabe que el método general está permutando los bits. Lo que no sabe es qué bit va en qué lugar, y esto es la clave.

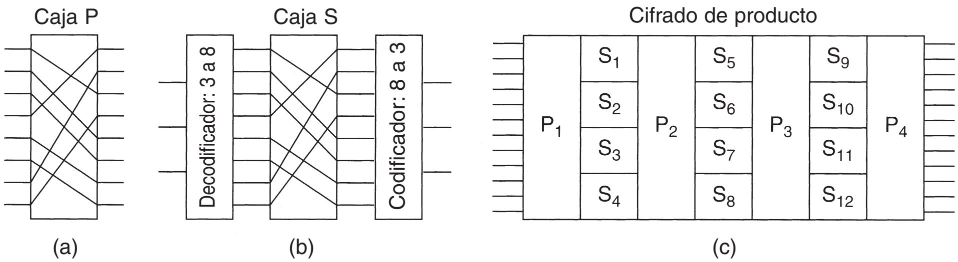


Figura 8-6. Elementos básicos del cifrado de producto. (a) Caja P. (b) Caja S. (c) Producto.

La sustitución se lleva a cabo mediante **cajas S,** como se muestra en la figura 8-6(b). En este ejemplo, se ingresa un texto llano de 3 bits y sale un texto cifrado de 3 bits. La entrada de 3 bits selecciona una de las ocho líneas de salida de la primera etapa y la establece en 1; las demás líneas son 0. La segunda etapa es una caja P. La tercera etapa codifica en binario nuevamente la línea de entrada seleccionada. Con el alambrado que se muestra, si los ocho números octales

01234567 entraran uno tras otro, la secuencia de salida sería de 24506713. En otras palabras, se ha reemplazado el 0 por el 2, el 1 por el 4, etcétera. Nuevamente, puede lograrse cualquier sustitución mediante el alambrado adecuado de la caja P dentro de la caja S. Además, tal dispositivo puede integrarse en hardware y puede alcanzarse una gran velocidad debido a que los codificadores y decodificadores sólo tienen uno o dos retardos de puerta lógica (subnanosegundos) y el tiempo de propagación a través de la caja P tal vez podría muy bien ser menor a 1 picosegundo.

La potencia real de estos elementos básicos sólo se hace aparente cuando ponemos en cascada una serie completa de cajas para formar un **cifrado de producto,** como se muestra en la figura 8-6(c). En este ejemplo, se trasponen (es decir, se permutan) 12 líneas de entrada en la primera etapa (P_1). En teoría, sería posible hacer que la segunda etapa fuera una caja S que relacionara un número de 12 bits con otro número de 12 bits. En vez de ello, tal dispositivo requeriría $2^{12} = 4096$ alambres cruzados en su etapa media. En cambio, la entrada se divide en cuatro grupos de 3 bits, cada uno de los cuales se sustituye independientemente de los demás. Aunque este método es menos general, aún es poderoso. Mediante la inclusión de una cantidad suficiente de etapas en el cifrado de producto, puede hacerse de la salida una función excesivamente complicada de la entrada.

Los cifrados de producto que operan en entradas de *k* bits para generar salidas de *k* bits son muy comunes. Por lo general, *k* es 64 a 256. Una implementación de hardware por lo general tiene por lo menos 18 etapas físicas, en lugar de sólo siete como en la figura 8-6(c). Una implementación de software se programa como un ciclo con por lo menos 8 iteraciones, cada una de las cuales realiza sustituciones de tipo caja S en subbloques del bloque de datos de 64 a 256 bits, seguido por una permutación que mezcla las salidas de las cajas S. Con frecuencia hay una permutación inicial especial y también una al final. En la literatura, las iteraciones se conocen como **rondas.**

8.2.1 DES—El Estándar de Encriptación de Datos

En enero de 1977, el gobierno de Estado Unidos adoptó un cifrado de producto desarrollado por IBM como su estándar oficial para información no secreta. Este cifrado, el **DES** (**Estándar de Encriptación de Datos**)**,** se adoptó ampliamente en la industria para usarse con productos de seguridad. Ya no es seguro en su forma original, pero aún es útil en una forma modificada. Ahora explicaremos el funcionamiento del DES.

En la figura 8-7(a) se muestra un esbozo del DES. El texto llano se encripta en bloques de 64 bits, produciendo 64 bits de texto cifrado. El algoritmo, que se parametriza mediante una clave de 56 bits, tiene 19 etapas diferentes. La primera etapa es una transposición, independiente de la clave, del texto llano de 64 bits. La última etapa es el inverso exacto de esta transposición. La etapa previa a la última intercambia los 32 bits de la izquierda y los 32 bits de la derecha. Las 16 etapas restantes son funcionalmente idénticas, pero se parametrizan mediante diferentes funciones de la clave. El algoritmo se ha diseñado para permitir que la desencriptación se haga con la misma clave que la encriptación. Los pasos simplemente se ejecutan en el orden inverso.

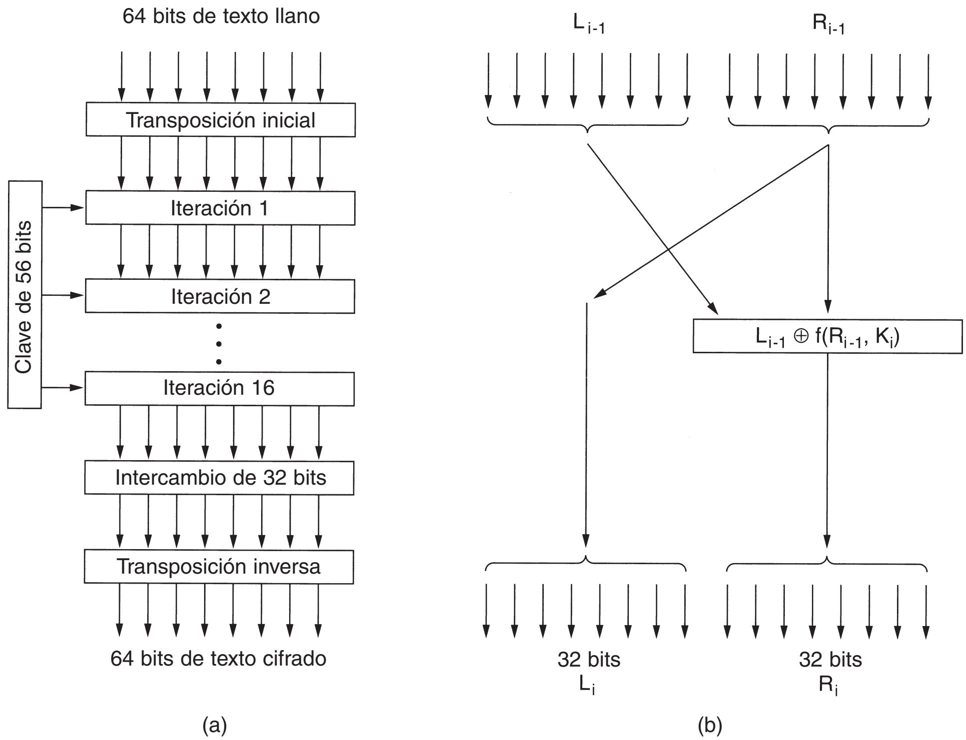


Figura 8-7. El estándar de encriptación de datos. (a) Esbozo general. (b) Detalle de una iteración. El signo de suma del círculo significa OR exclusivo.

La operación de una de estas etapas intermedias se ilustra en la figura 8-7(b). Cada etapa toma dos entradas de 32 bits y produce dos salidas de 32 bits. La salida de la izquierda simplemente es una copia de la entrada de la derecha. La salida de la derecha es el OR exclusivo a nivel de bits de la entrada izquierda y una función de la entrada derecha y la clave de esta etapa, K_i. Toda la complejidad reside en esta función.

La función consiste en cuatro pasos, ejecutados en secuencia. Primero se construye un número de 48 bits, E, expandiendo el R_{i-1} de 32 bits según una regla fija de transposición y duplicación. Después, se aplica un OR exclusivo a E y K_i. Esta salida entonces se divide en ocho grupos de 6 bits, cada uno de los cuales se alimenta a una caja S distinta. Cada una de las 64 entradas posibles a la caja S se mapea en una salida de 4 bits. Por último, estos 8×4 bits se pasan a través de una caja P.

En cada una de las 16 iteraciones, se usa una clave diferente. Antes de iniciarse el algoritmo, se aplica una transposición de 56 bits a la clave. Justo antes de cada iteración, la clave se divide en dos unidades de 28 bits, cada una de las cuales se gira hacia la izquierda una cantidad de bits dependiente del número de iteración. K_i se deriva de esta clave girada aplicándole otra transposición de 56 bits. En cada vuelta se extrae y permuta de los 56 bits un subgrupo de 48 bits diferente.

Una técnica que algunas veces se utiliza para hacer a DES más fuerte se conoce como **blanqueamiento** (*whitening*). Consiste en aplicar un OR exclusivo a una clave aleatoria de 64 bits con cada bloque de texto llano antes de alimentarla al DES y después aplicar un OR exclusivo a una segunda clave de 64 bits con el texto cifrado resultante antes de transmitirla. El blanqueamiento puede eliminarse fácilmente mediante la ejecución de las operaciones inversas (si el receptor tiene las dos claves de blanqueamiento). Puesto que esta técnica agrega más bits a la longitud de la clave, provoca que una búsqueda completa del espacio de claves consuma mucho más tiempo. Observe que se utiliza la misma clave de blanqueamiento para cada bloque (es decir, sólo hay una clave de blanqueamiento).

El DES se ha visto envuelto en controversias desde el día en que se lanzó. Se basó en un cifrado desarrollado y patentado por IBM, llamado Lucifer, excepto que dicho cifrado usaba una clave de 128 bits en lugar de una de 56 bits. Cuado el gobierno federal de Estados Unidos quiso estandarizar un método de cifrado para uso no confidencial, "invitó" a IBM a "debatir" el asunto con la NSA, la dependencia de descifrado de códigos del gobierno, que es el empleador de matemáticos y criptólogos más grande del mundo. La NSA es tan secreta que una broma de la industria reza:

Pregunta: ¿Qué significa NSA?
Respuesta: Ninguna Supuesta Agencia.

En realidad, NSA significa Agencia Nacional de Seguridad de Estados Unidos.

Tras estos debates, IBM redujo la clave de 128 a 56 bits y decidió mantener en secreto el proceso de diseño del DES. Mucha gente sospechó que la longitud de la clave se redujo para asegurar que la NSA pudiera descifrar el código, pero no ninguna organización de menor presupuesto. El objetivo del diseño secreto supuestamente era esconder una puerta trasera que pudiera facilitar aún más a la NSA el descifrado del DES. Cuando un empleado de la NSA pidió discretamente al IEEE que cancelara una conferencia planeada sobre criptografía, eso incomodó aún más a la gente. La NSA negó todo.

En 1977, dos investigadores de criptografía de Stanford, Diffie y Hellman (1977) diseñaron una máquina para violar el DES y estimaron que el costo de la construcción podría ascender a unos 20 millones de dólares. Dado un trozo pequeño de texto llano y el texto cifrado correspondiente, esta máquina podría encontrar la clave mediante una búsqueda exhaustiva del espacio de claves de 2^{56} entradas en menos de un día. Hoy día tal máquina costaría tal vez menos de un millón de dólares.

Triple DES

Ya en 1979, IBM se dio cuenta de que la longitud de la clave DES era muy corta y diseñó una forma de incrementarla de manera efectiva, utilizando cifrado triple (Tuchman, 1979). El método elegido, que desde entonces se ha incorporado en el Estándar Internacional 8732, se ilustra en la figura 8-8. Aquí se utilizan dos claves y tres etapas. En la primera etapa, el texto llano se encripta mediante DES de la forma usual con K_1. En la segunda etapa, DES se ejecuta en modo de desencriptación, utilizando K_2 como la clave. Por último, se realiza otra encriptación DES con K_1.

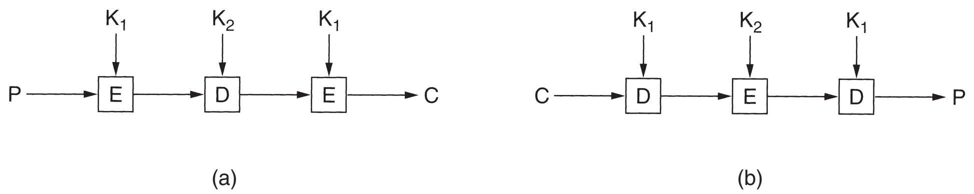


Figura 8-8. (a) Triple encriptación con DES. (b) Desencriptación.

Este diseño da lugar inmediatamente a dos preguntas. Primera, ¿por qué sólo se usan dos claves en lugar de tres? Segunda, ¿por qué se utiliza **EDE** (**Encriptar Desencriptar Encriptar**) en lugar de **EEE** (**Encriptar Encriptar Encriptar**)? La razón de que se usen dos claves es que incluso los criptógrafos más paranoicos coinciden en que por ahora 112 bits son suficientes para las aplicaciones comerciales. (Y entre los criptógrafos la paranoia se considera una ventaja, no un error.) Subir a 168 bits simplemente agregaría la sobrecarga innecesaria de administrar y transportar otra clave.

La razón para encriptar, desencriptar y luego encriptar de nuevo es la compatibilidad hacia atrás con los sistemas DES de una sola clave. Tanto las funciones de encriptación como de desencriptación son correspondencias entre grupos de números de 64 bits. Desde el punto de vista criptográfico, las dos correspondencias son igualmente robustas. Sin embargo, mediante el uso de EDE en lugar de EEE, una computadora que emplea triple encriptación puede comunicarse con otra que usa encriptación sencilla simplemente estableciendo $K_1 = K_2$. Esta propiedad permite la introducción gradual de la triple encriptación, algo que no interesa a los criptógrafos académicos, pero que es de importancia considerable para IBM y sus clientes.

8.2.2 AES—El Estándar de Encriptación Avanzada

Conforme el DES comenzó a acercarse al final de su vida útil, aun con el DES triple, el **NIST** (**Instituto Nacional de Estándares y Tecnología),** la agencia del Departamento de Comercio de Estados Unidos encargada de aprobar estándares del Gobierno Federal de Estados Unidos, decidió que el gobierno necesitaba un nuevo estándar criptográfico para uso no confidencial. El NIST estaba completamente consciente de toda la controversia alrededor del DES y sabía que si sólo anunciaba un nuevo estándar, todos los que tuvieran conocimiento sobre criptografía supondrían de manera automática que la NSA había construido una puerta trasera con el fin de leer todo lo que se encriptara con ella. Bajo estas condiciones, probablemente nadie utilizaría el estándar y con seguridad tendría una muerte silenciosa.

Por esto, el NIST adoptó una estrategia sorprendentemente diferente para una burocracia gubernamental: promovió un concurso. En enero de 1997, los investigadores de todo el mundo fueron invitados a emitir propuestas para un nuevo estándar, que se llamaría **AES** (**Estándar de Encriptación Avanzada**)**.** Las reglas fueron:

1. El algoritmo debe ser un cifrado de bloques simétricos.
2. Todo el diseño debe ser público.
3. Deben soportarse las longitudes de claves de 128, 192 y 256 bits.
4. Deben ser posibles las implementaciones tanto de software como de hardware.
5. El algoritmo debe ser público o con licencia en términos no discriminatorios.

Se realizaron quince propuestas serias y se organizaron conferencias para presentarlas, en las cuales se alentó activamente a los asistentes a que encontraran errores en todas ellas. En agosto de 1998, el NIST seleccionó cinco finalistas con base en su seguridad, eficiencia, simplicidad, flexibilidad y requerimientos de memoria (importantes para los sistemas integrados). Se llevaron a cabo más conferencias y se tomaron más críticas. En la última conferencia se realizó una votación no obligatoria. Los finalistas y sus puntajes fueron los siguientes:

1. Rijndael (de Joan Daemen y Vincent Rijmen, 86 votos).
2. Serpent (de Ross Anderson, Eli Biham y Lars Knudsen, 59 votos).
3. Twofish (de un equipo encabezado por Bruce Schneier, 31 votos).
4. RC6 (de los Laboratorios RSA, 23 votos).
5. MARS (de IBM, 13 votos).

En octubre de 2000, el NIST anunció que él también votó por Rijndael, y en noviembre de 2001 Rijndael se volvió un estándar del gobierno de Estados Unidos publicado como FIPS 197 (Estándar Federal para el Procesamiento de Información). Debido a la extraordinaria apertura de la competencia, las propiedades técnicas de Rijndael y al hecho de que el equipo ganador estuvo compuesto por dos jóvenes criptógrafos belgas (quienes no es probable que hayan construido una puerta trasera sólo para complacer a la NSA), se espera que Rijndael se vuelva el estándar criptográfico dominante en el mundo por lo menos por una década. El nombre Rijndael se deriva de los apellidos de los autores: Rijmen + Daemen.

Rijndael soporta longitudes de clave y tamaños de bloque de 128 a 256 bits en pasos de 32 bits. Las longitudes de clave y de bloque pueden elegirse de manera independiente. Sin embargo, el AES especifica que el tamaño de bloque debe ser de 128 bits y la longitud de clave debe ser de 128, 192 o 256 bits. Es indudable que nadie utilizará claves de 192 bits, por lo que, de hecho, el AES tiene dos variantes: un bloque de 128 bits con clave de 128 bits y un bloque de 128 bits con clave de 256 bits.

En el tratamiento de nuestro siguiente algoritmo examinaremos sólo el caso 128/128 porque es probable que éste se vuelva la norma comercial. Una clave de 128 bits da un espacio de claves de $2^{128} \approx 3 \times 10^{38}$ claves. Incluso si la NSA se las arregla para construir una máquina con mil millones de procesadores paralelos, cada uno de los cuales es capaz de evaluar una clave por picosegundo, a tal máquina le llevaría 10^{10} años buscar en el espacio de claves. Para ese entonces el Sol

ya se habrá apagado, por lo que las personas existentes tendrán que leer los resultados a la luz de las velas.

Rijndael

Desde una perspectiva matemática, Rijndael se basa en la teoría de campos de Galois, la cual da algunas propiedades verificables de seguridad. Sin embargo, también puede verse como código C, sin meterse a las matemáticas.

Al igual que el DES, Rijndael utiliza sustitución y permutaciones, así como múltiples rondas. El número de rondas depende del tamaño de clave y del tamaño de bloque, y es de 10 para las claves de 128 bits con bloques de 128 bits y aumenta hasta 14 para la clave o el bloque más grande. Sin embargo, a diferencia del DES, todas las operaciones involucran bytes completos, para permitir implementaciones eficientes tanto en hardware como en software. En la figura 8-9 se muestra un esquema del código.

```
#define LENGTH 16                        /* # de bytes en el bloque de datos o en
                                            la clave */
#define NROWS 4                          /* número de filas en estado */
#define NCOLS 4                          /* número de columnas en estado */
#define ROUNDS 10                        /* número de iteraciones */
typedef unsigned char byte;              /* entero sin signo de 8 bits */

rijndael(byte plaintext[LENGTH], byte ciphertext[LENGTH], byte key[LENGTH])
{
 int r;                                  /* índice de ciclo */
 byte state[NROWS][NCOLS];               /* estado actual */
 struct {byte k[NROWS][NCOLS];} rk[ROUNDS + 1];  /* claves de ronda */

 expand_key(key, rk);                    /* construye las claves de ronda */
 copy_plaintext_to_state(state, plaintext); /* inicia el estado actual */
 xor_roundkey_into_state(state, rk[0]); /* aplica OR exclusivo a la clave dentro
                                             del estado */

 for (r = 1; r <= ROUNDS; r++) {
     substitute(state);                  /* aplica la caja S a cada byte */
     rotate_rows(state);                 /* gira la fila i en i bytes */
     if (r < ROUNDS) mix_columns(state); /* función de mezcla */
     xor_roundkey_into_state(state, rk[r]);  /* aplica OR exclusivo a la clave
                                                 dentro del estado */
 }
 copy_state_to_ciphertext(ciphertext, state);    /* devuelve el resultado */
}
```

Figura 8-9. Un esquema de Rijndael.

La función *rijndael* tiene tres parámetros: *plaintext*, un arreglo de 16 bytes que contiene los datos de entrada; *ciphertext,* un arreglo de 16 bytes a donde se regresará la salida cifrada, y *key*, la clave de 16 bytes. Durante el cálculo, el estado actual de los datos se mantiene en un arreglo de

bytes, *state*, cuyo tamaño es *NROWS* × *NCOLS*. Para bloques de 128 bits, este arreglo es de 4 × 4 bytes. Con 16 bytes, se puede almacenar el bloque de datos completo de 128 bits.

El arreglo *state* se inicializa al texto llano y se modifica en cada paso en el cálculo. En algunos pasos, se realiza la sustitución de byte por byte. En otros, los bytes se permutan dentro del arreglo. También se utilizan otras transformaciones. Al final, el contenido de *state* se regresa como texto cifrado.

El código inicia expandiendo la clave en 11 arreglos del mismo tamaño que el estado. Éstos se almacenan en *rk*, que es un arreglo de estructuras, cada una de las cuales contiene un arreglo de estado. Uno de los arreglos se utilizará al inicio del cálculo y los otros 10 se utilizarán en 10 rondas, uno por ronda. El cálculo de las claves de ronda de la clave de encriptación es muy complicado para tratarlo aquí. Basta decir que las claves de ronda se producen mediante una rotación repetida y aplicando OR exclusivo a varios grupos de bits de clave. Para todos los detalles, vea (Daemen y Rijmen, 2002).

El siguiente paso es copiar el texto llano en el arreglo *state* a fin de poder procesarlo durante las rondas. Se copia en orden de columna, con los primeros cuatro bytes colocados en la columna 0, los siguientes cuatro bytes en la columna 1, y así sucesivamente. Las columnas y las filas se numeran comenzando en 0, aunque las rondas se numeren comenzando en 1. En la figura 8-10 se muestra la configuración inicial de los 12 arreglos de bytes de tamaño 4 × 4.

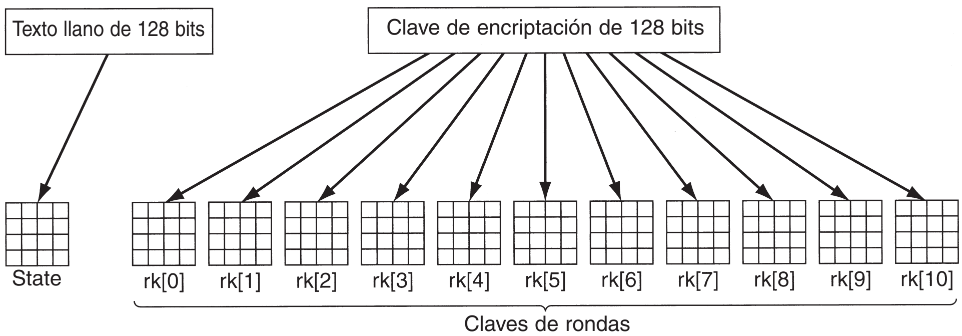


Figura 8-10. Creación de los arreglos *state* y *rk*.

Hay un paso adicional antes de que comience el cálculo principal: se aplica OR exclusivo a *rk*[0] dentro de *state*, byte por byte. En otras palabras, cada uno de los 16 bytes de *state* se reemplaza con el resultado del OR exclusivo de sí mismo y el byte correspondiente de *rk*[0].

Es el momento de presentar la atracción principal. El ciclo ejecuta 10 iteraciones, una por ronda, y transforma a *state* en cada iteración. El contenido de cada ronda consiste de cuatro pasos. El paso 1 realiza una sustitución de byte por byte sobre *state*. Cada byte se utiliza a la vez como un índice en una caja S para reemplazar su valor por el contenido de entrada de la caja S. Este paso es un cifrado de sustitución monoalfabética directo. A diferencia del DES, que tiene múltiples cajas S, Rijndael sólo tiene una caja S.

El paso 2 gira a la izquierda cada una de las cuatro filas. La fila 0 se gira 0 bytes (es decir, no cambia), la fila 1 se gira 1 byte, la fila 2 se gira 2 bytes y la fila 3 se gira 3 bytes. Este paso disemina por el bloque el contenido de los datos actuales, de forma análoga a las permutaciones de la figura 8-6.

El paso 3 mezcla cada una de las columnas independientemente de las demás. La mezcla se realiza utilizando multiplicación de matriz en la que cada nueva columna es el producto de la columna vieja (antes de la multiplicación) y una matriz constante, y la multiplicación se realiza mediante el campo finito de Galois, $GF(2^8)$. Aunque esto podría sonar complicado, existe un algoritmo que permite calcular cada elemento de la nueva columna mediante dos búsquedas de tabla y tres OR exclusivos (Daemen y Rijmen, 2002, apéndice E).

Por último, el paso 4 aplica OR exclusivo a la clave de esta ronda dentro del arreglo *state.*

Puesto que cada paso es reversible, la desencriptación se puede realizar con sólo ejecutar el algoritmo en sentido inverso. Sin embargo, también hay un truco mediante el cual la desencriptación se puede realizar ejecutando el algoritmo de encriptación y utilizando tablas diferentes.

El algoritmo se ha diseñado no sólo para una gran seguridad, sino también para una gran velocidad. Una buena implementación de software en una máquina de 2 GHz debe tener la capacidad de alcanzar una tasa de encriptación de 700 Mbps, que es lo suficientemente rápida para encriptar cerca de 100 vídeos MPEG-2 en tiempo real. Las implementaciones de hardware son aún más rápidas.

8.2.3 Modos de cifrado

A pesar de toda esta complejidad, el AES (o el DES o, de hecho, cualquier cifrado de bloques) es básicamente un cifrado de sustitución monoalfabética que utiliza caracteres grandes (caracteres de 128 bits para el AES y caracteres de 64 bits para el DES). Siempre que el mismo bloque de texto llano entra en la etapa inicial, el mismo bloque de texto cifrado sale de la etapa final. Si encripta 100 veces el texto llano *abcdefgh* con la misma clave DES, obtiene 100 veces el mismo texto cifrado. Un intruso puede aprovechar esta propiedad para violar el cifrado.

Modo de libro de código electrónico

Para ver cómo puede utilizarse esta propiedad de cifrado de sustitución monoalfabética para vencer parcialmente el cifrado, utilizaremos el (triple) DES porque es más fácil diseñar bloques de 64 bits que de 128 bits, pero el AES tiene exactamente el mismo problema. La forma directa de utilizar el DES para cifrar una pieza grande de texto llano es dividirla en bloques consecutivos de 8 bytes (64 bits) y encriptarlos después uno tras otro con la misma clave. La última pieza de texto llano se rellena a 64 bits, en caso de ser necesario. Esta técnica se conoce como **modo ECB** (**modo de Libro de Código Electrónico**) en analogía con los libros de código pasados de moda en los que se listaba cada palabra de texto llano, seguida por su texto cifrado (por lo general, un número decimal de cinco dígitos).

En la figura 8-11 se muestra el inicio de un archivo de computadora que lista los bonos anuales que una compañía ha decidido otorgar a sus empleados. Este archivo consiste en registros consecutivos de 32 bytes, uno por empleado, en el formato que se muestra: 16 bytes para el nombre,

8 bytes para el puesto y 8 bytes para el bono. Cada uno de los 16 bloques de 8 bytes (numerados del 0 al 15) se encripta mediante (triple) DES.

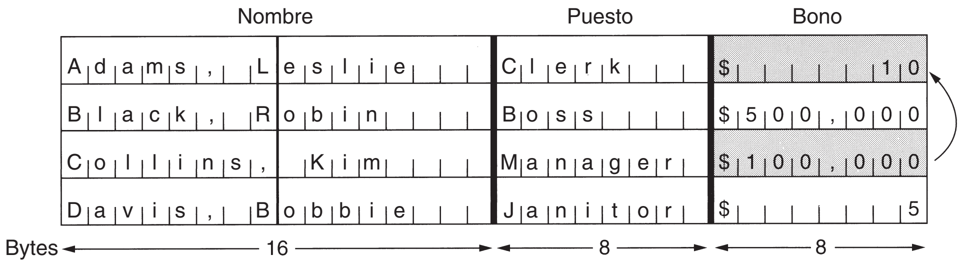


Figura 8-11. El texto llano de un archivo encriptado como 16 bloques DES.

Leslie tiene una pelea con el jefe y no espera un bono significativo. Kim, en contraste, es la favorita del jefe, y todos saben esto. Leslie puede obtener acceso al archivo después de que se encripta, pero antes de que se envíe al banco. ¿Leslie puede enmendar esta situación injusta con sólo contar con el archivo encriptado?

No hay problema. Todo lo que Leslie tiene que hacer es realizar una copia del doceavo bloque de texto cifrado (que contiene el bono de Kim) y utilizarlo para reemplazar el cuarto bloque de texto cifrado (que contiene el bono de Leslie). Incluso sin saber lo que dice el doceavo bloque, Leslie puede esperar contar con una Navidad mucho más feliz este año. (Copiar el octavo bloque de texto cifrado también es una posibilidad, pero es más probable de detectar; además, Leslie no es una persona avara.)

Modo de encadenamiento de bloques de cifrado

Para frustrar este tipo de ataque, todos los cifrados en bloques pueden encadenarse de varias formas a fin de que el reemplazo de un bloque de la forma en que lo hizo Leslie cause que el texto llano se desencripte comenzando en el bloque reemplazado que se desechará. Una forma de encadenar es el **encadenamiento de bloques de cifrado.** En este método, que se muestra en la figura 8-12, a cada bloque de texto llano se le aplica un OR exclusivo con el bloque anterior de texto cifrado antes de ser encriptado. En consecuencia, el mismo bloque de texto llano ya no corresponde con el mismo bloque de texto cifrado, y la encriptación deja de ser un enorme cifrado de sustitución monoalfabética. Al primer bloque se le aplica un OR exclusivo con un **IV** (**Vector de Inicialización**) elegido de manera aleatoria, que se transmite (en texto llano) junto con el texto cifrado.

En el ejemplo de la figura 8-12 podemos ver la forma en que funciona el modo de encadenamiento de bloques de cifrado. Iniciamos calculando $C_0 = E(P_0 \text{ XOR } IV)$. Después calculamos $C_1 = E(P_1 \text{ XOR } C_0)$, y así sucesivamente. La desencriptación también utiliza OR exclusivo para invertir el proceso, con $P_0 = IV \text{ XOR } D(C_0)$, etcétera. Observe que la encriptación del bloque i es una función de todo el texto llano de los bloques 0 a $i - 1$, por lo que el mismo texto llano genera texto cifrado diferente dependiendo de dónde ocurre. Una transformación del tipo que realizó Leslie no tendrá sentido para dos bloques que inician en el campo de bono de Leslie. Para un oficial de seguridad astuto, esta peculiaridad podría sugerir en dónde comenzar la investigación resultante.

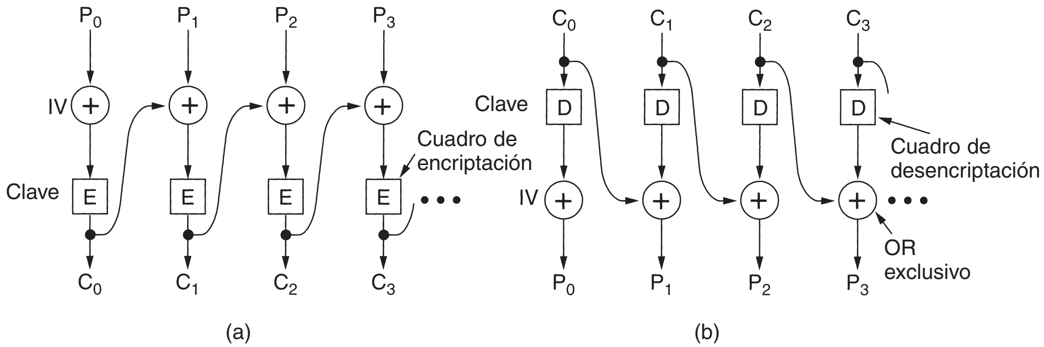


Figura 8-12. Encadenamiento de bloques de cifrado. (a) Encriptación. (b) Desencriptación.

El encadenamiento de bloques cifrado también tiene la ventaja de que el mismo bloque de texto llano no resultará en el mismo bloque de texto cifrado, lo que dificulta el criptoanálisis. De hecho, ésta es la razón principal por la que se utiliza.

Modo de retroalimentación de cifrado

Sin embargo, el encadenamiento de bloques tiene la desventaja de que requiere la llegada de un bloque completo de 64 bits antes de que pueda comenzar la desencriptación. Este modo no es adecuado para terminales interactivas, en las que se pueden escribir líneas máximo de ocho caracteres y es necesario detenerse a esperar una respuesta. Para la encriptación byte por byte, **modo de retroalimentación de cifrado,** se utiliza (triple) DES, como se muestra en la figura 8-13. Para el AES la idea es exactamente la misma; sólo se utiliza un registro de desplazamiento de 128 bits. En esta figura se muestra el estado de la máquina de encriptación después de que se han encriptado y enviado los bytes 0 a 9. Cuando llega el byte 10 de texto llano, como se ilustra en la figura 8-13(a), el algoritmo DES opera en el registro de desplazamiento de 64 bits para generar texto cifrado de 64 bits. El byte más a la izquierda de ese texto cifrado se extrae y se le aplica un OR exclusivo con P_{10}. Ese byte se transmite en la línea de transmisión. Además, el registro de desplazamiento se mueve a la izquierda 8 bits, causando que C_2 se caiga del extremo izquierdo y que C_{10} se inserte en la posición que C_9 acaba de dejar libre en el extremo derecho. Observe que el contenido del registro de desplazamiento depende de la historia completa anterior del texto llano, por lo que un patrón que se repite múltiples veces en el texto llano se encriptará de manera diferente cada vez en el texto cifrado. Al igual que con el encadenamiento de bloques de cifrado, se necesita un vector de inicialización para comenzar.

La desencriptación con el modo de retroalimentación de cifrado hace lo mismo que la encriptación. En particular, el contenido del registro de desplazamiento se *encripta*, no se *desencripta*, a fin de que el byte seleccionado al cual se le aplica el OR exclusivo con C_{10} para obtener P_{10} sea el mismo al que se le aplicó el OR exclusivo con P_{10} para generar C_{10} en primera instancia. Siempre y cuando los dos registros de desplazamiento permanezcan idénticos, la desencriptación funcionará de manera correcta. Esto se ilustra en la figura 8-13(b).

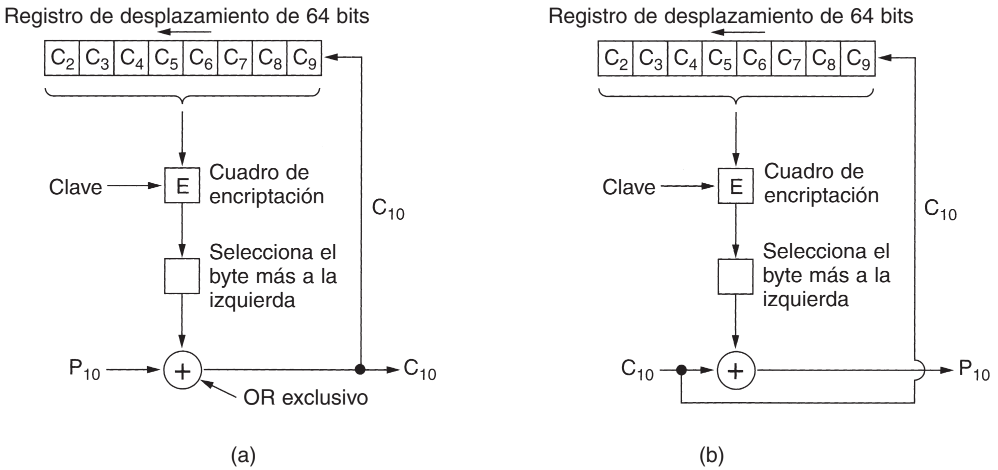


Figura 8-13. Modo de retroalimentación de cifrado. (a) Encriptación. (b) Desencriptación.

Un problema con el modo de retroalimentación de cifrado es que si un bit del texto cifrado se invierte de manera accidental durante la transmisión, se dañarán los 8 bytes que se desencriptan mientras el byte incorrecto se encuentra en el registro de desplazamiento. Una vez que el byte incorrecto se elimine del registro de desplazamiento, se generará nuevamente texto llano correcto. Por lo tanto, los efectos de un solo bit invertido se limitan relativamente a un sitio y no se daña el resto del mensaje, pero sí se dañan los bits que haya en el registro de desplazamiento.

Modo de cifrado de flujo

Sin embargo, existen aplicaciones en las que un error de transmisión de 1 bit que arruina 64 bits de texto llano es demasiado. Para estas aplicaciones, existe una cuarta opción, el **modo de cifrado de flujo.** Funciona encriptando un vector de inicialización y usando una clave para obtener un bloque de salida. A continuación se encripta este bloque usando la clave para obtener un segundo bloque de salida. A continuación este bloque se encripta para obtener un tercer bloque, y así sucesivamente. La secuencia (arbitrariamente grande) de bloques de salida, llamada **flujo de claves,** se trata como un relleno de una sola vez y se le aplica OR exclusivo con el texto llano para obtener el texto cifrado, como se muestra en la figura 8-14(a). Observe que el IV se utiliza sólo en el primer paso. Después de eso, se encripta la salida. Observe también que el flujo de claves es independiente de los datos, por lo que puede calcularse por adelantado, si es necesario, y es completamente insensible a los errores de transmisión. En la figura 8-14(b) se muestra la desencriptación.

La desencriptación se realiza generando el mismo flujo de claves en el lado receptor. Puesto que el flujo de claves depende sólo del IV y de la clave, no le afectan los errores de transmisión en el texto cifrado. Por lo tanto, un error de 1 bit en el texto cifrado transmitido genera sólo un error de 1 bit en el texto llano desencriptado.

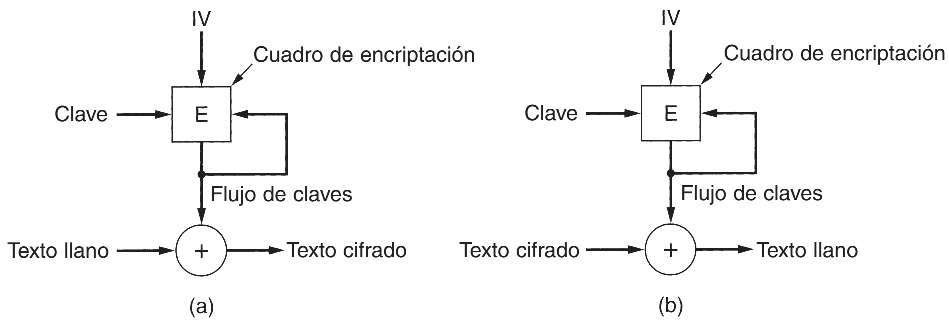


Figura 8-14. Un cifrado de flujo. (a) Encriptación. (b) Desencriptación.

Es esencial nunca utilizar dos veces el mismo par (clave, IV) con un cifrado de flujo, pues de hacerlo generará cada vez el mismo flujo de claves. Utilizar dos veces el mismo flujo de claves expone al texto cifrado a un **ataque de reutilización de flujo de claves.** Imagine que el bloque de texto llano, P_0, se encripta con el flujo de claves para obtener P_0 XOR K_0. Posteriormente, un segundo bloque de texto llano, Q_0, se encripta con el mismo flujo de claves para obtener Q_0 XOR K_0. Un intruso que capture estos dos bloques de texto cifrado puede sencillamente aplicarles un OR exclusivo en conjunto para obtener P_0 XOR Q_0, lo cual elimina la clave. Ahora el intruso tiene el OR exclusivo de los dos bloques de texto llano. Si se conoce alguno de ellos o es posible adivinarlo, el otro también se puede descubrir. En cualquier caso, el OR exclusivo de dos flujos de texto llano puede ser atacado utilizando propiedades estadísticas del mensaje. Por ejemplo, para texto en inglés, tal vez el carácter más común en el flujo será el OR exclusivo de dos espacios, seguido por el OR exclusivo de un espacio y la letra "e", etc. En resumen, al poseer el OR exclusivo de dos bloques de texto llano, el criptoanalista tiene una excelente oportunidad de adivinarlos.

Modo de contador

Un problema que todos los modos tienen, excepto el modo de libro de código electrónico, es que el acceso aleatorio a datos encriptados es imposible. Por ejemplo, suponga que un archivo se transmite a través de una red y después se almacena en disco de forma encriptada. Ésta sería una forma razonable de operar si la computadora receptora fuera una computadora portátil con riesgo de ser robada. Almacenar todos los archivos importantes de forma encriptada reduce en gran medida el potencial daño que se podría sufrir por la información confidencial que se fugaría en caso de que la computadora cayera en las manos equivocadas.

Sin embargo, los archivos de disco con frecuencia se acceden en orden no secuencial, especialmente los archivos de bases de datos. Con un archivo encriptado mediante encadenamiento de bloques de cifrado, el acceso a un bloque aleatorio requiere que primero se desencripten todos los bloques que estén delante de él, lo cual es muy costoso. Por esta razón, se ha inventado otro modo, el **modo de contador,** como se ilustra en la figura 8-15. Aquí el texto llano no se encripta en forma

directa. En su lugar, se encripta el vector de inicialización más una constante, y al texto cifrado resultante se le aplica un OR exclusivo con el texto llano. Al incrementar en 1 el vector de inicialización por cada nuevo bloque, es más fácil desencriptar un bloque en cualquier parte del archivo sin tener que desencriptar primero todos sus predecesores.

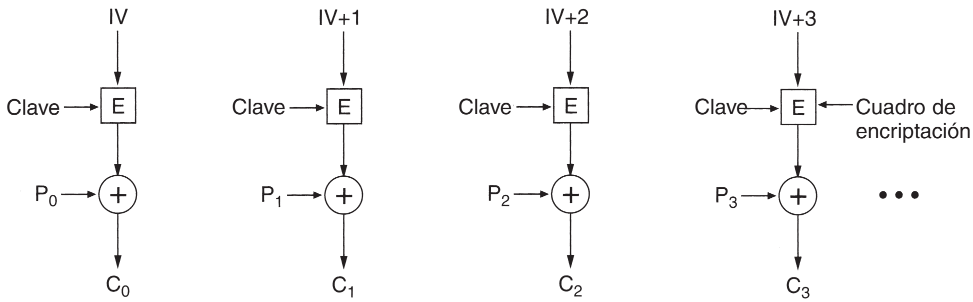


Figura 8-15. Encriptación con el modo de contador.

Aunque el modo de contador es útil, tiene una debilidad que vale la pena mencionar. Suponga que en el futuro se utiliza nuevamente la misma clave, *K* (con texto llano diferente pero el mismo IV), y un atacante adquiere todo el texto cifrado de ambas corridas. Los flujos de claves son los mismos en los dos casos, y exponen el cifrado a un ataque de reutilización de flujo de claves del mismo tipo que vimos en los cifrados de flujo. Todo lo que el criptoanalista tiene que hacer es aplicar un OR exclusivo a los dos textos cifrados en conjunto para eliminar toda la protección criptográfica y sólo obtener el OR exclusivo de los textos llanos. Esta debilidad no significa que el modo de contador sea una mala idea. Simplemente significa que las claves y los vectores de inicialización deben elegirse de manera independiente y al azar. Aun cuando la misma clave se utilice dos veces de manera accidental, si el IV es diferente cada vez, el texto llano estará a salvo.

8.2.4 Otros cifrados

DES y Rijndael son los algoritmos criptográficos de clave simétrica más conocidos. Sin embargo, vale la pena mencionar que se han diseñado otros cifrados de clave simétrica. Algunos de ellos están incluidos en varios productos. En la figura 8-16 se listan algunos de los más comunes.

8.2.5 Criptoanálisis

Antes de dejar el tema de la criptografía de clave simétrica, vale la pena cuando menos mencionar cuatro avances recientes del criptoanálisis. El primero es el **criptoanálisis diferencial** (Biham y Shamir, 1993). Esta técnica puede utilizarse para atacar cualquier cifrado en bloques; empieza con un par de bloques de texto llano que difieren sólo en una cantidad pequeña de bits y observando cuidadosamente lo que ocurre en cada iteración interna a medida que avanza la

Cifrado	Autor	Longitud de clave	Comentarios
Blowfish	Bruce Schneier	1–448 bits	Antiguo y lento
DES	IBM	56 bits	Muy débil para utilizarlo en la actualidad
IDEA	Massey y Xuejia	128 bits	Bueno, pero patentado
RC4	Ronald Rivest	1–2048 bits	Precaución: algunas claves son débiles
RC5	Ronald Rivest	128–256 bits	Bueno, pero patentado
Rijndael	Daemen y Rijmen	128–256 bits	La mejor opción
Serpent	Anderson, Biham, Knudsen	128–256 bits	Muy robusto
Triple DES	IBM	168 bits	Segunda mejor opción
Twofish	Bruce Schneier	128–256 bits	Muy robusto; ampliamente utilizado

Figura 8-16. Algunos algoritmos criptográficos comunes de clave simétrica.

encriptación. En muchos casos, algunos patrones son mucho más comunes que otros, y esta observación conduce a un ataque probabilístico.

El segundo avance que vale la pena mencionar es el **criptoanálisis lineal** (Matsui, 1994). Éste puede descifrar el DES con sólo 2^{43} textos llanos conocidos. Funciona aplicando un OR exclusivo a ciertos bits del texto llano y el texto cifrado en conjunto y buscando patrones en el resultado. Al hacerse repetidamente, la mitad de los bits deben ser 0s y la otra mitad 1s. Sin embargo, con frecuencia los cifrados introducen una desviación en una dirección o en la otra, y esta desviación, por pequeña que sea, puede explotarse para reducir el factor de trabajo. Para los detalles, vea el trabajo de Matsui.

El tercer avance es el análisis del consumo de energía eléctrica para averiguar las claves secretas. Las computadoras por lo general utilizan 3 voltios para representar un bit 1 y 0 voltios para representar un bit 0. Por lo tanto, procesar un 1 gasta más energía eléctrica que procesar un 0. Si un algoritmo criptográfico consiste en un ciclo en el que los bits clave se procesan en orden, un atacante que reemplace el reloj principal de *n* GHz con uno lento (por ejemplo, 100 Hz) y coloque pinzas de caimán en los pines de energía y tierra de la CPU, puede monitorear con precisión la energía consumida por cada instrucción de la máquina. A partir de estos datos, deducir la clave es sorprendentemente fácil. Este tipo de criptoanálisis puede vencerse sólo al codificar con sumo cuidado el algoritmo en lenguaje ensamblador para asegurarse de que el consumo de energía sea independiente de la clave, así como de todas las claves de rondas individuales.

El cuarto avance es el análisis de temporización. Los algoritmos criptográficos están llenos de instrucciones `if` que prueban bits en las claves de ronda. Si las partes `then` y `else` toman diferentes cantidades de tiempo, reduciendo la velocidad del reloj y viendo el tiempo que tardan en ejecutarse varios pasos, también podría ser posible deducir las claves de ronda. Una vez que se conocen todas las claves de ronda, por lo general puede calcularse la clave original. Los análisis de energía y temporización también pueden utilizarse de manera simultánea para facilitar el trabajo. Si bien los análisis de energía y temporización pueden parecer exóticos, en realidad son técnicas poderosas que pueden romper cualquier cifrado que no esté específicamente diseñado para resistirlas.

8.3 ALGORITMOS DE CLAVE PÚBLICA

Históricamente el problema de distribución de claves siempre ha sido la parte débil de la mayoría de los criptosistemas. Sin importar lo robusto que sea un criptosistema, si un intruso puede robar la clave, el sistema no vale nada. Los criptólogos siempre daban por hecho que las claves de encriptación y desencriptación eran la misma (o que se podía derivar fácilmente una de la otra). Pero la clave tenía que distribuirse a todos los usuarios del sistema. Por lo tanto, parecía haber un problema inherente: las claves se tenían que proteger contra robo, pero también se tenían que distribuir, por lo que no podían simplemente guardarse en una caja fuerte.

En 1976, dos investigadores de la Universidad de Stanford, Diffie y Hellman (1976), propusieron una clase nueva de criptosistema, en el que las claves de encriptación y desencriptación eran diferentes y la clave de desencriptación no podía derivarse de la clave de encriptación. En su propuesta, el algoritmo de encriptación (con clave), E, y el algoritmo de desencriptación (con clave), D, tenían que cumplir con los tres requisitos siguientes. Estos requisitos pueden expresarse sencillamente como sigue:

1. $D(E(P)) = P$.
2. Es excesivamente difícil deducir D de E.
3. E no puede descifrarse mediante un ataque de texto llano seleccionado.

El primer requisito dice que, si aplicamos D a un mensaje cifrado, $E(P)$, obtenemos nuevamente el mensaje de texto llano original, P. Sin esta propiedad, el receptor legítimo no podría desencriptar el texto cifrado. El segundo requisito no requiere explicación. El tercer requisito es necesario porque, como veremos en un momento, los intrusos pueden experimentar a placer con el algoritmo. En estas condiciones, no hay razón para que una clave de encriptación no pueda hacerse pública.

El método funciona como sigue. Una persona, llamémosla Alice, que quiera recibir mensajes secretos, primero diseña dos algoritmos, E_A y D_A, que cumplan los requisitos anteriores. El algoritmo de encriptación y la clave de Alice se hacen públicos, de ahí el nombre de **criptografía de clave pública.** Por ejemplo, Alice podría poner su clave pública en su página de inicio en Web. Utilizaremos la notación E_A para denotar el algoritmo de encriptación parametrizado por la clave pública de Alice. De manera similar, el algoritmo de desencriptación (secreto) parametrizado por la clave privada de Alice es D_A. Bob hace lo mismo, haciendo pública E_B pero manteniendo secreta D_B.

Ahora veamos si podemos resolver el problema de establecer un canal seguro entre Alice y Bob, que nunca han tenido contacto previo. Se supone que tanto la clave de encriptación de Alice, E_A, como la clave de encriptación de Bob, E_B, están en un archivo de lectura pública. Ahora, Alice toma su primer mensaje, P, calcula $E_B(P)$ y lo envía a Bob. Bob entonces lo desencripta aplicando su clave secreta D_B [es decir, calcula $D_B(E_B(P)) = P$]. Nadie más puede leer el mensaje

encriptado, $E_B(P)$, porque se supone que el sistema de encriptación es robusto y porque es demasiado difícil derivar D_B de la E_B públicamente conocida. Para enviar una respuesta, R, Bob transmite $E_A(R)$. Ahora, Alice y Bob se pueden comunicar con seguridad.

Es útil una nota sobre terminología. La criptografía de clave pública requiere que cada usuario tenga dos claves: una clave pública, usada por todo el mundo para encriptar mensajes a enviar a ese usuario, y una clave privada, que necesita el usuario para desencriptar los mensajes. Consistentemente nos referimos a estas claves como claves *públicas* y *privadas*, respectivamente, y las distinguiremos de las claves *secretas* usadas en la criptografía convencional de clave simétrica.

8.3.1 El algoritmo RSA

La única dificultad estriba en que necesitamos encontrar algoritmos que realmente satisfagan estos tres requisitos. Debido a las ventajas potenciales de la criptografía de clave pública, muchos investigadores están trabajando a todo vapor, y ya se han publicado algunos algoritmos. Un buen método fue descubierto por un grupo del M.I.T. (Rivest y cols., 1978). Es conocido por las iniciales de sus tres descubridores (Rivest, Shamir, Adleman): **RSA.** Ha sobrevivido a todos los intentos para romperlo por más de un cuarto de siglo y se le considera muy robusto. Mucha de la seguridad práctica se basa en él. Su mayor desventaja es que requiere claves de por lo menos 1024 bits para una buena seguridad (en comparación con los 128 bits de los algoritmos de clave simétrica), por lo cual es muy lento.

Su método se basa en ciertos principios de la teoría de los números. Ahora resumiremos el uso del método; para los detalles, consulte el trabajo original.

1. Seleccionar dos números primos grandes, p y q (generalmente de 1024 bits).
2. Calcular $n = p \times q$ y $z = (p - 1) \times (q - 1)$.
3. Seleccionar un número primo con respecto a z, llamándolo d.
4. Encontrar e tal que $e \times d = 1 \ mod \ z$.

Con estos parámetros calculados por adelantado, estamos listos para comenzar la encriptación. Dividimos el texto llano (considerado como una cadena de bits) en bloques, para que cada mensaje de texto llano, P, caiga en el intervalo $0 \leq P < n$. Esto puede hacerse agrupando el texto llano en bloques de k bits, donde k es el entero más grande para el que $2^k < n$ es verdad.

Para encriptar un mensaje, P, calculamos $C = P^e$ (mod n). Para desencriptar C, calculamos $P = C^d$ (mod n). Puede demostrarse que, para todos los P del intervalo especificado, las funciones de encriptación y desencriptación son inversas. Para ejecutar la encriptación, se necesitan e y n. Para llevar a cabo la desencriptación, se requieren d y n. Por tanto, la clave pública consiste en el par (e, n), y la clave privada consiste en (d, n).

La seguridad del método se basa en la dificultad para factorizar números grandes. Si el criptoanalista pudiera factorizar n (conocido públicamene), podría encontrar p y q y, a partir de éstos, z. Equipado con el conocimiento de z y de e, puede encontrar d usando el algoritmo

de Euclides. Afortunadamente, los matemáticos han estado tratando de factorizar números grandes durante los últimos 300 años, y las pruebas acumuladas sugieren que se trata de un problema excesivamente difícil.

De acuerdo con Rivest y sus colegas, la factorización de un número de 500 dígitos requiere 10^{25} años de tiempo de cómputo utilizando el mejor algoritmo conocido y una computadora con un tiempo de instrucción de 1 μseg. Aun si las computadoras continúan aumentando su velocidad en un orden de magnitud cada década, pasarán siglos antes de que sea factible la factorización de un número de 500 dígitos, y para entonces nuestros descendientes simplemente pueden escoger un p y un q todavía más grandes.

Un ejemplo pedagógico trivial del algoritmo RSA se muestra en la figura 8-17. Para este ejemplo hemos seleccionado $p = 3$ y $q = 11$, dando $n = 33$ y $z = 20$. Un valor adecuado de d es $d = 7$, puesto que 7 y 20 no tienen factores comunes. Con estas selecciones, e puede encontrarse resolviendo la ecuación $7e = 1 \pmod{20}$, que produce $e = 3$. El texto cifrado, C, de un mensaje de texto llano, P, se da por la regla $C = P^3 \pmod{33}$. El receptor desencripta el texto cifrado de acuerdo con la regla $P = C^7 \pmod{33}$. En la figura se muestra como ejemplo la encriptación del texto llano "SUZANNE".

Texto llano (P)			Texto cifrado (C)		Después de la desencriptación	
Simbólico	Numérico	P^3	P^3 (mod 33)	C^7	C^7 (mod33)	Simbólico
S	19	6859	28	13492928512	19	S
U	21	9261	21	1801088541	21	U
Z	26	17576	20	1280000000	26	Z
A	01	1	1	1	01	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	05	E
Cálculo del emisor				Cálculo del receptor		

Figura 8-17. Ejemplo del algoritmo RSA.

Dado que los números primos escogidos para este ejemplo son tan pequeños, P debe ser menor que 33, por lo que cada bloque de texto llano puede contener sólo un carácter. El resultado es un cifrado por sustitución monoalfabética, no muy impresionante. En cambio, si hubiéramos seleccionado p y $q \approx 2^{512}$, podríamos tener $n \approx 2^{1024}$, de tal manera que cada bloque podría ser de hasta 1024 bits o 128 caracteres de 8 bits, en comparación con 8 caracteres para el DES y 16 para el AES.

Cabe señalar que el uso del RSA como lo hemos descrito es semejante a usar un algoritmo simétrico en modo ECB: el mismo bloque de entrada da el mismo bloque de salida. Por tanto, se requiere alguna forma de encadenamiento para la encriptación de datos. Sin embargo, en la práctica la mayoría de los sistemas basados en RSA usan criptografía de clave pública principalmente para distribuir claves de sesión de una sola vez para su uso con algún algoritmo de clave simétrica

como el AES o el triple DES. El RSA es demasiado lento para poder encriptar grandes volúmenes de datos, pero se utiliza con amplitud para la distribución de claves.

8.3.2 Otros algoritmos de clave pública

Aunque el RSA se usa ampliamente, de ninguna manera es el único algoritmo de clave pública conocido. El primer algoritmo de clave pública fue el de la mochila (Merkle y Hellman, 1978). La idea aquí es que alguien es dueño de una gran cantidad de objetos, todos con pesos diferentes. El dueño cifra el mensaje seleccionando secretamente un subgrupo de los objetos y colocándolos en la mochila. El peso total de los objetos contenidos en la mochila se hace público, así como la lista de todos los objetos posibles. La lista de los objetos que se metieron en la mochila se mantiene en secreto. Con ciertas restricciones adicionales, el problema de determinar una lista posible de los objetos a partir del peso dado se consideró no computable, y formó la base del algoritmo de clave pública.

El inventor del algoritmo, Ralph Merkle, estaba bastante seguro de que este algoritmo no podía violarse, por lo que ofreció una recompensa de 100 dólares a cualquiera que pudiera descifrarlo. Adi Shamir (la "S" de RSA) pronto lo violó y cobró la recompensa. Sin desmotivarse, Merkle reforzó el algoritmo y ofreció una recompensa de 1000 dólares a quien pudiera violar el nuevo. Ronald Rivest (la "R" de RSA) pronto lo descifró y cobró la recompensa. Merkle no se atrevió a ofrecer 10,000 dólares para la siguiente versión, por lo que "A" (Leonard Adleman) no tuvo suerte. Aunque se ha modificado nuevamente, el algoritmo de la mochila no se considera seguro y pocas veces se usa.

Otros esquemas de clave pública se basan en la dificultad para calcular logaritmos discretos. El Gamal (1985) y Schnorr (1991) han inventado algoritmos basados en este principio.

Existen algunos otros esquemas, como los basados en curvas elípticas (Menezes y Vanstone, 1993), pero las dos categorías principales son las basadas en la dificultad para factorizar números grandes y calcular logaritmos discretos módulo un número primo grande. Estos problemas se consideran verdaderamente difíciles de resolver porque los matemáticos han estado trabajando en ellos durante años sin adelantos importantes.

8.4 FIRMAS DIGITALES

La autenticidad de muchos documentos legales, financieros y de otros tipos se determina por la presencia o ausencia de una firma manuscrita autorizada. Las fotocopias no cuentan. Para que los sistemas de mensajes computarizados reemplacen el transporte físico de papel y tinta, debe encontrarse un método para que la firma de los documentos sea infalsificable.

El problema de inventar un reemplazo para las firmas manuscritas es difícil. Básicamente, lo que se requiere es un sistema mediante el cual una parte pueda enviar un mensaje "firmado" a otra parte de modo que:

1. El receptor pueda verificar la identidad del transmisor.
2. El transmisor no pueda repudiar (negar) después el contenido del mensaje.
3. El receptor no haya podido elaborar el mensaje él mismo.

El primer requisito es necesario, por ejemplo, en los sistemas financieros. Cuando la computadora de un cliente ordena a la computadora de un banco que compre una tonelada de oro, la computadora del banco necesita asegurarse de que la computadora que da la orden realmente pertenece a la compañía a la que se le aplicará el débito. En otras palabras, el banco tiene que autenticar la identidad del cliente (y éste a su vez tiene que hacer lo propio).

El segundo requisito es necesario para proteger al banco contra fraude. Supongamos que el banco compra la tonelada de oro, e inmediatamente después cae marcadamente el precio del oro. Un cliente deshonesto podría demandar al banco, alegando que nunca emitió una orden para comprar el oro. Cuando el banco presenta un mensaje en la corte, el cliente niega haberlo enviado. La propiedad de que ninguna parte de un contrato pueda negar haber firmado se conoce como **no repudio.** Los esquemas de firma digital que analizaremos ayudan a proporcionar dicha propiedad.

El tercer requisito es necesario para proteger al cliente en el caso de que el precio del oro suba mucho y que el banco trate de falsificar un mensaje firmado en el que el cliente solicitó un lingote de oro en lugar de una tonelada. En este escenario fraudulento, el banco simplemente mantiene el resto del oro para sí mismo.

8.4.1 Firmas de clave simétrica

Un enfoque de las firmas digitales sería tener una autoridad central que sepa todo y en quien todos confíen, digamos el *Big Brother* (*BB*). Cada usuario escoge entonces una clave secreta y la lleva personalmente a las oficinas del *BB*. Por tanto, sólo Alice y el *BB* conocen la clave secreta de Alice, K_A, etcétera.

Cuando Alice quiere enviar un mensaje de texto llano firmado, *P*, a su banquero, Bob, genera $K_A(B, R_A, t, P)$ donde *B* es la identidad de Bob, R_A es un número aleatorio elegido por Alice, *t* es una marca de tiempo para asegurar que el mensaje sea reciente, y $K_A(B, R_A, t, P)$ es el mensaje encriptado con su clave, K_A. A continuación lo envía como se muestra en la figura 8-18. El *BB* ve que el mensaje es de Alice, lo desencripta, y envía un mensaje a Bob como se muestra. El mensaje a Bob contiene el texto llano del mensaje de Alice y también el mensaje firmado $K_{BB}(A, t, P)$. Ahora, Bob atiende la solicitud de Alice.

¿Qué ocurre si Alice niega posteriormente haber enviado el mensaje? El paso 1 es que todos demandan a todos (cuando menos en Estados Unidos). Por último, cuando el caso llega a la corte y Alice niega haber enviado a Bob el mensaje en disputa, el juez pregunta a Bob por qué está tan seguro de que el mensaje en disputa vino de Alice y no de Trudy. Bob primero indica que el *BB* no aceptaría un mensaje de Alice a menos que estuviera encriptado con K_A, por lo que no hay posibilidad de que Trudy enviara al *BB* un mensaje falso de Alice sin que el *BB* lo detectara de inmediato.

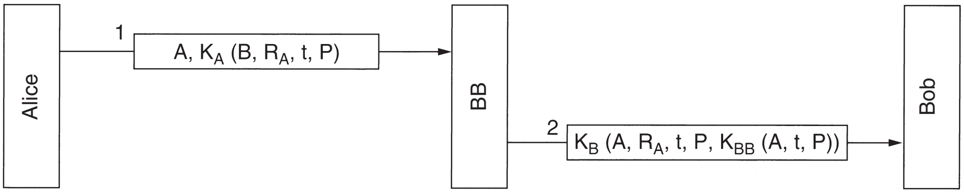


Figura 8-18. Firmas digitales con el Big Brother.

Bob entonces presenta la prueba A, K_{BB} (*A, t, P*). Bob dice que éste es un mensaje firmado por el *BB* que comprueba que Alice envió *P* a Bob. El juez entonces pide al *BB* (en quien todo el mundo confía) que descifre la prueba A. Cuando el *BB* testifica que Bob dice la verdad, el juez se pronuncia a favor de Bob. Caso cerrado.

Un problema potencial del protocolo de firma de la figura 8-18 es que Trudy repita cualquiera de los dos mensajes. Para minimizar este problema, en todos los intercambios se usan marcas de tiempo. Es más, Bob puede revisar todos los mensajes recientes para ver si se usó R_A en cualquiera de ellos. De ser así, el mensaje se descarta como repetición. Observe que Bob rechazará los mensajes muy viejos con base en la marca de tiempo. Para protegerse contra ataques de repetición instantánea, Bob simplemente examina el R_A de cada mensaje de entrada para ver si un mensaje igual se recibió de Alice durante la hora pasada. Si no, Bob puede suponer con seguridad que éste es una solicitud nueva.

8.4.2 Firmas de clave pública

Un problema estructural del uso de la criptografía de clave simétrica para las firmas digitales es que todos tienen que confiar en el *Big Brother*. Es más, el *Big Brother* lee todos los mensajes firmados. Los candidatos más lógicos para operar el servidor del *Big Brother* son el gobierno, los bancos, los contadores y los abogados. Por desgracia, ninguna de estas organizaciones inspira confianza completa a todos los ciudadanos. Por tanto, sería bueno si la firma de documentos no requiriese una autoridad confiable.

Afortunadamente, la criptografía de clave pública puede hacer una contribución importante aquí. Supongamos que los algoritmos públicos de encriptación y desencriptación tienen la propiedad de que $E(D(P)) = P$ además de la propiedad normal de $D(E(P)) = P$. (El RSA tiene esta propiedad, por lo que el supuesto es razonable.) Suponiendo que éste es el caso, Alice puede enviar un mensaje de texto llano firmado, *P*, a Bob, transmitiendo $E_B(D_A(P))$. Observe que Alice conoce su propia clave (privada), D_A, así como la clave pública de Bob, E_B, por lo que Alice puede elaborar este mensaje.

Cuando Bob recibe el mensaje, lo transforma usando su clave privada, como es normal, produciendo $D_A(P)$, como se muestra en la figura 8-19. Bob almacena este texto en un lugar seguro y lo descifra usando E_A para obtener el texto llano original.

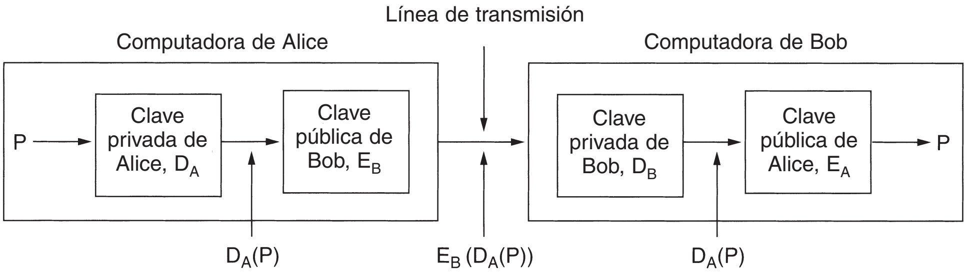


Figura 8-19. Firmas digitales usando criptografía de clave pública.

Para ver cómo funciona la propiedad de firma, supongamos que Alice posteriormente niega haber enviado el mensaje P a Bob. Cuando el caso llega a la corte, Bob puede presentar tanto P como $D_A(P)$. El juez puede comprobar fácilmente que Bob tiene un mensaje válido encriptado con D_A con sólo aplicarle E_A. Puesto que Bob no conoce la clave privada de Alice, la única forma en que Bob pudo haber adquirido un mensaje encriptado con esa clave sería que Alice en efecto lo hubiera enviado. Mientras está en la cárcel por perjurio y fraude, Alice tendrá tiempo suficiente para diseñar algoritmos de clave pública nuevos e interesantes.

Aunque el uso de la criptografía de clave pública para las firmas digitales es un esquema elegante, hay problemas relacionados con el entorno en el que opera más que con el algoritmo básico. Por una parte, Bob puede demostrar que un mensaje fue enviado por Alice siempre y cuando D_A permanezca en secreto. Si Alice divulga su clave secreta, el argumento ya no es válido, puesto que cualquiera pudo haber enviado el mensaje, incluido el mismo Bob.

El problema podría surgir, por ejemplo, si Bob es el corredor de bolsa de Alice. Alice le indica a Bob que compre ciertas acciones o bonos. Inmediatamente después, el precio cae en picada. Para negar haberle enviado el mensaje a Bob, Alice corre a la policía para informarles que robaron su casa y que también sustrajeron su clave. Dependiendo de las leyes de su estado o país, puede o no ser responsable legalmente, en especial si indica no haber descubierto el robo sino hasta después de llegar del trabajo, varias horas después.

Otro problema con el esquema de firmas es qué ocurre si Alice decide cambiar su clave. Hacerlo ciertamente es legal, y probablemente es una buena idea cambiar la clave periódicamente. Si luego surge un caos en la corte, como se describió antes, el juez aplicará la E_A *actual* a $D_A(P)$ y descubrirá que no produce P. Bob quedará en ridículo en ese momento.

En principio, cualquier algoritmo de clave pública puede usarse para firmas digitales. El estándar *de facto* de la industria es el algoritmo RSA, y muchos productos de seguridad lo usan. Sin embargo, en 1991, el NIST (Instituto Nacional de Estándares y Tecnología) propuso el uso de una variación del algoritmo de clave pública de El Gamal para su nuevo **Estándar de Firmas Digitales** (**DSS**). La seguridad de El Gamal radica en la dificultad para calcular logaritmos discretos, en lugar de la dificultad para factorizar números grandes.

Como es normal cuando el gobierno intenta dictar estándares criptográficos, hubo una protesta general. El DSS tuvo críticas por ser

1. Demasiado secreto (el NSA diseñó el protocolo para usar El Gamal).
2. Demasiado lento (10 a 40 veces más lento que el RSA para comprobar firmas).
3. Demasiado nuevo (El Gamal no ha sido analizado exhaustivamente).
4. Demasiado inseguro (clave fija de 512 bits).

En una modificación posterior, el cuarto punto se prestó a discusión pública cuando se permitieron claves de hasta 1024 bits. Sin embargo, los primeros dos puntos permanecen válidos.

8.4.3 Compendios de mensaje

Una crítica a los métodos de firma es que con frecuencia combinan dos funciones dispares: autenticación y confidencialidad. En muchos casos se requiere la autenticación, pero no confidencialidad. Asimismo, con frecuencia la obtención de una licencia de exportación se facilita si el sistema en cuestión sólo proporciona autenticación pero no confidencialidad. A continuación describiremos un esquema de autenticación que no requiere la encriptación del mensaje completo.

Este esquema se base en la idea de una función de *hash* unidireccional que toma una parte arbitrariamente grande de texto llano y a partir de ella calcula una cadena de bits de longitud fija. Esta función de *hash*, *MD*, llamada **compendio de mensaje** (*message digest*), tiene cuatro propiedades importantes:

1. Dado P, es fácil calcular $MD(P)$.
2. Dado $MD(P)$, es imposible encontrar P.
3. Dado P nadie puede encontrar P' de tal manera que $MD(P') = MD(P)$.
4. Un cambio a la entrada de incluso 1 bit produce una salida muy diferente.

Para cumplir el criterio 3, la función de *hash* debe ser de cuando menos 128 bits de longitud, y de preferencia mayor. Para cumplir el criterio 4, la función de *hash* debe truncar los bits minuciosamente, de manera semejante a como lo hacen los algoritmos de encriptación de clave simétrica que hemos visto.

El cálculo de un compendio de mensaje a partir de un trozo de texto llano es mucho más rápido que la encriptación de ese texto llano con un algoritmo de clave pública, por lo que los compendios de mensaje pueden usarse para acelerar los algoritmos de firma digital. Para ver su funcionamiento, considere nuevamente el protocolo de firma de la figura 8-18. En lugar de firmar P con $K_{BB}(A, t, P)$, el *BB* ahora calcula el compendio del mensaje aplicando *MD* a P para producir $MD(P)$. El *BB* entonces incluye $K_{BB}(A, t, MD(P))$ como quinto elemento de la lista encriptada con K_B que se envía a Bob, en lugar de $K_{BB}(A, t, P)$.

Si surge una disputa, Bob puede presentar tanto P como $K_{BB}(A, t, MD(P))$. Una vez que el *Big Brother* lo desencripta para el juez, Bob tiene $MD(P)$, que está garantizado que es genuino, y el P

supuesto. Dado que es prácticamente imposible que Bob encuentre otro mensaje que dé este resultado, el juez se convencerá fácilmente de que Bob dice la verdad. Este uso de compendios de mensaje ahorra tanto tiempo de encriptación como costos de transporte de mensajes.

Los compendios de mensaje funcionan también en los criptosistemas de clave pública, como se muestra en la figura 8-20. Aquí, Alice primero calcula el compendio de mensaje de su texto llano: luego firma el compendio del mensaje y envía a Bob tanto el compendio firmado como el texto llano. Si Trudy reemplaza *P* en el camino, Bob verá esto cuando calcule *MD*(*P*) él mismo.

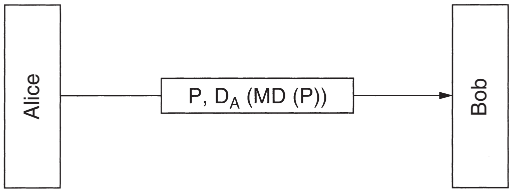


Figura 8-20. Firmas digitales usando compendios de mensaje.

MD5

Se ha propuesto una variedad de funciones para el compendio de mensajes. Las de mayor uso son MD5 (Rivest, 1992) y SHA-1 (NIST, 1993). **MD5** es la quinta de una serie de compendios de mensaje diseñados por Ronald Rivest. Opera truncando los bits de una manera tan complicada que cada bit de salida es afectado por cada bit de entrada. Muy brevemente, comienza por rellenar el mensaje a una longitud de 448 bits (módulo 512). Después la longitud original del mensaje se agrega como entero de 64 bits para dar una entrada total cuya longitud es un múltiplo de 512 bits. El último paso del cálculo previo es la inicialización de un búfer de 128 bits a un valor fijo.

Ahora comienza el cálculo. Cada ronda toma un bloque de 512 bits de entrada y lo mezcla por completo con el búfer de 128 bits. Por si fuera poco, se introduce también una tabla construida a partir de la función seno. El objetivo de usar una función conocida como el seno no es porque sea más aleatoria que un generador de números aleatorios, sino para evitar cualquier sospecha de que el diseñador construyó una puerta trasera ingeniosa por la que sólo él puede entrar. La negativa de IBM a hacer públicos los principios en que se basó el diseño de las cajas S del DES dio pie a una gran cantidad de especulación sobre las puertas traseras. Se hacen cuatro rondas por cada bloque de entrada. Este proceso continúa hasta que todos los bloques de entrada se han consumido. El contenido del búfer de 128 bits forma el compendio del mensaje.

MD5 ha existido aproximadamente por una década, y muchas personas lo han atacado. Se han encontrado algunas vulnerabilidades, pero ciertos pasos internos evitan que sea violado. Sin embargo, si cayeran las barreras restantes dentro de MD5, éste podría fallar con el tiempo. Sin embargo, al momento de escribir esto, aún están de pie.

SHA-1

La otra función principal para el compendio de mensajes es **SHA-1** (**Algoritmo Seguro de *Hash* 1),** desarrollado por NSA y aprobado por el NIST en FIPS 180-1. Al igual que MD5, SHA-1 procesa datos de entrada en bloques de 512 bits, sólo a diferencia de MD5, genera un compendio de mensaje de 160 bits. En la figura 8-21 se ilustra una forma típica para que Alice envíe a Bob un mensaje no secreto, pero firmado. Aquí su mensaje de texto llano se alimenta en el algoritmo SHA-1 para obtener un *hash* SHA-1 de 160 bits. A continuación Alice firma el *hash* con su clave privada RSA y envía a Bob tanto el mensaje de texto llano como el *hash* firmado.

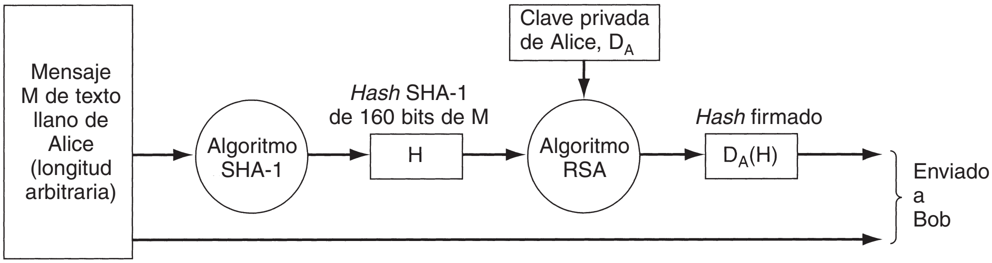


Figura 8-21. Uso de SHA-1 y RSA para firmar mensajes no secretos.

Después de recibir el mensaje, Bob calcula el *hash* SHA-1 él mismo y también aplica la clave pública de Alice al *hash* firmado para obtener el *hash* original, *H*. Si los dos concuerdan, el mensaje se considera válido. Puesto que no hay forma de que Trudy modifique el mensaje (de texto llano) mientras está en tránsito y producir uno nuevo que haga *hash* a *H*, Bob puede detectar con facilidad cualquier cambio que Trudy haya hecho al mensaje. Para los mensajes cuya integridad es importante pero cuyo contenido no es secreto, se utiliza ampliamente el esquema de la figura 8-21. Por un costo de cómputo relativamente bajo, garantiza que cualquier modificación hecha al mensaje de texto llano en tránsito pueda detectarse con una probabilidad muy alta.

Ahora veamos brevemente cómo funciona SHA-1. Comienza rellenando el mensaje con 1 bit al final, seguido de tantos bits 0 como sean necesarios para que la longitud sea un múltiplo de 512 bits. A continuación, al número de 64 bits que contiene la longitud del mensaje antes del relleno se le aplica un OR dentro de los 64 bits de menor orden. En la figura 8-22, el mensaje se muestra con un relleno a la derecha debido a que el texto y las figuras van de izquierda a derecha (es decir, el extremo inferior derecho por lo general se percibe como el final de la figura). En las computadoras esta orientación corresponde a máquinas *big-endian* como la SPARC, pero SHA-1 siempre rellena el final del mensaje, sin importar cuál máquina *endian* se utilice.

Durante el cálculo, SHA-1 mantiene variables de 32 bits, H_0 a H_4, donde se acumula el *hash*. Dichas variables se muestran en la figura 8-22(b). Se inicializan a constantes especificadas en el estándar.

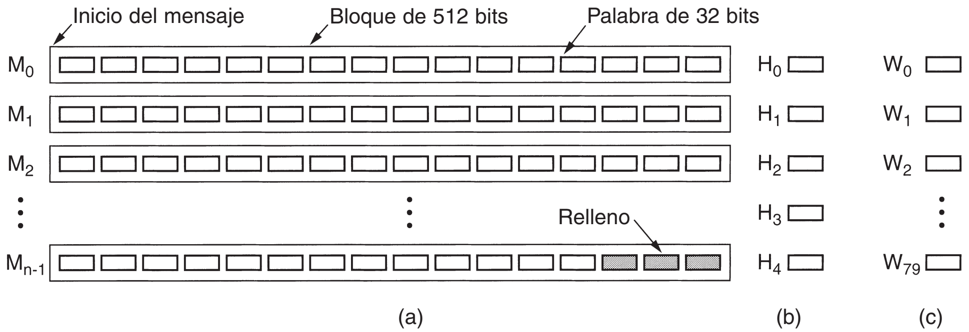


Figura 8-22. (a) Un mensaje rellenado a un múltiplo de 512 bits. (b) Las variables de salida. (c) El arreglo de palabras.

Ahora se procesa cada uno de los bloques M_0 a M_{n-1}. Para el bloque actual, las 16 palabras primero se copian al inicio de un arreglo auxiliar de 80 palabras, *W*, como se muestra en la figura 8-22(c). Después las otras 64 palabras de *W* se rellenan utilizando la fórmula

$$W_i = S^1\,(W_{i-3} \text{ XOR } W_{i-8} \text{ XOR } W_{i-14} \text{ XOR } W_{i-16}) \qquad (16 \leq i \leq 79)$$

donde $S^b(W)$ representa la rotación circular izquierda de la palabra de 32 bits, *W*, por *b* bits. Ahora cinco variables de trabajo, *A* a *E* se inicializan de H_0 a H_4, respectivamente.

El cálculo real puede expresarse en pseudo-C como

```
for (i = 0; i < 80; i++) {
     temp = S5 (A) + fi (B, C, D) + E + Wi + Ki;
     E = D; D = C; C = S30(B); B = A; A = temp;
}
```

donde las constantes K_i se definen en el estándar. Las funciones de mezcla f_i se definen como

$$f_i\,(B,C,D) = (B \text{ AND } C) \text{ OR } (\text{NOT } B \text{ AND } D) \qquad (0 \leq i \leq 19)$$
$$f_i\,(B,C,D) = B \text{ XOR } C \text{ XOR } D \qquad (20 \leq i \leq 39)$$
$$f_i\,(B,C,D) = (B \text{ AND } C) \text{ OR } (B \text{ AND } D) \text{ OR } (C \text{ AND } D) \qquad (40 \leq i \leq 59)$$
$$f_i\,(B,C,D) = B \text{ XOR } C \text{ XOR } D \qquad (60 \leq i \leq 79)$$

Cuando las 80 iteraciones del ciclo estén completas, *A* a *E* se agregan a H_0 a H_4, respectivamente.

Ahora que se han procesado los primeros bloques de 512 bits, se inicia el siguiente. El arreglo *W* se reinicia desde el nuevo bloque, pero *H* se deja como estaba. Cuando se termina este bloque, se inicia el siguiente, y así sucesivamente, hasta que los bloques de mensajes de 512 bits se hayan procesado por completo. Cuando se termina el último bloque, las cinco palabras de 32 bits en el arreglo *H* se envían a la salida como el *hash* criptográfico de 160 bits. El código C completo para SHA-1 se da en el RFC 3174.

Nuevas versiones de SHA-1 están en desarrollo para *hashes* de 256, 384 y 512 bits, respectivamente.

8.4.4 El ataque de cumpleaños

En el mundo de la criptografía, nada es lo que parece. Podríamos pensar que se requieren del orden de 2^m operaciones para subvertir un compendio de mensajes de m bits. De hecho, con frecuencia $2^{m/2}$ operaciones son suficientes si se usa el **ataque de cumpleaños,** un enfoque publicado por Yuval (1979) en su ahora clásico trabajo "*How to Swindle Rabin*" (Cómo estafar a Rabin).

La idea de este ataque proviene de una técnica que con frecuencia usan los profesores de matemáticas en sus cursos de probabilidad. La pregunta es: ¿Cuántos estudiantes se necesitan en un grupo antes de que la probabilidad de tener dos personas con el mismo cumpleaños exceda 1/2? Muchos estudiantes suponen que la respuesta debe ser mucho mayor que 100. De hecho, la teoría de la probabilidad indica que es de apenas 23. Sin hacer un análisis riguroso, intuitivamente, con 23 personas, podemos formar $(23 \times 22)/2 = 253$ pares diferentes, cada uno de los cuales tiene una probabilidad de 1/365 de cumplir el requisito. Bajo esta luz, la respuesta ya no es en realidad tan sorprendente.

En términos más generales, si hay alguna correspondencia entre las entradas y las salidas con n entradas (gente, mensajes, etc.) y k salidas posibles (cumpleaños, compendios de mensajes, etc.) hay $n(n-1)/2$ pares de entradas. Si $n(n-1)/2 > k$, la posibilidad de que cuando menos una coincida es bastante buena. Por lo tanto, en términos aproximados, es probable una igualación para $n > \sqrt{k}$. Este resultado significa que un compendio de mensajes de 64 bits probablemente puede violarse generando unos 2^{32} mensajes y buscando dos con el mismo compendio de mensaje.

Veamos ahora un ejemplo práctico. El Departamento de Informática de la Universidad Estatal tiene una cátedra para un miembro facultativo y tiene dos candidatos, Tom y Dick. Tom fue contratado dos años antes que Dick, por lo que su candidatura será considerada antes. Si Tom obtiene el puesto, mala suerte para Dick. Tom sabe que la jefa del departamento, Marilyn, tiene buen concepto de su trabajo, por lo que le pide que escriba una carta de recomendación para el rector, quien decidirá el caso de Tom. Una vez enviadas, todas las cartas se vuelven confidenciales.

Marilyn le dice a su secretaria, Ellen, que escriba una carta al rector, delineando lo que quiere en ella. Cuando está lista, Marilyn la revisará, calculará y firmará el compendio de 64 bits y lo enviará al rector. Ellen puede mandar la carta después por correo electrónico.

Desgraciadamente para Tom, Ellen tiene una relación sentimental con Dick y le gustaría dejar fuera a Tom, por lo que escribe la carta siguiente con las 32 opciones entre corchetes.

Estimado rector Smith,

Esta [*carta* | *mensaje*] es para dar mi opinión [*franca* | *honesta*] sobre el profesor Tom Wilson, que [*ahora* | *este año*] [*es candidato a* | *está en espera de*] una cátedra. He [*conocido a* | *trabajado con*] el profesor Wilson durante [*unos* | *casi*] seis años. Es un investigador [*sobresaliente* | *excelente*] de gran [*talento* | *habilidad*] conocido [*mundialmente* | *internacionalmente*] por sus [*brillantes* | *creativas*] investigaciones sobre [*muchos* | *una gran variedad de*] problemas [*difíciles* | *desafiantes*].

Él es también un [*profesor* | *educador*] [*altamente* | *grandemente*] [*respetado* | *admirado*]. Sus estudiantes han hecho evaluaciones [*sobresalientes* | *espectaculares*] de sus [*clases* | *cursos*]; es el [*profesor* | *instructor*] [*más admirado* | *más querido*] [*del departamento* | *por nosotros*].

[*Además* | *Por otra parte*], el profesor Wilson es [*hábil* | *diligente*] para obtener financiamiento. Sus [*subvenciones* | *contratos*] han aportado una cantidad [*importante* | *sustancial*] de dinero [*al* | *a nuestro*] departamento. [*Este dinero ha* | *Estos fondos han*] [*posibilitado* | *permitido*] que [*emprendamos* | *pongamos en práctica*] muchos programas [*especiales* | *importantes*], [*como* | *por ejemplo*] su programa Estado 2000. Sin estos fondos [*seríamos incapaces de* | *no podríamos*] continuar este programa, que es tan [*importante* | *esencial*] para ambos. Recomiendo encarecidamente que se le otorgue la cátedra.

Desgraciadamente para Tom, tan pronto como Ellen termina de redactar y mecanografiar esta carta, también escribe una segunda:

Estimado rector Smith:

Esta [*carta* | *mensaje*] es para dar mi opinión [*franca* | *honesta*] sobre el profesor Tom Wilson, que [*ahora* | *este año*] [*es candidato a* | *está en espera de*] una cátedra. He [*conocido a* | *trabajado con*] el profesor Wilson durante [*unos* | *casi*] seis años. Es un investigador [*malo* | *mediocre*] poco conocido en su [*campo* | *área*]. Sus investigaciones [*casi nunca* | *pocas veces*] muestran [*entendimiento* | *comprensión del*] los problemas [*clave* | *principales*] [*del día* | *de nuestros tiempos*].

Es más, no es un [*profesor* | *educador*] [*respetado* | *admirado*]. Sus estudiantes han hecho evaluaciones [*pobres* | *malas*] de sus [*clases* | *cursos*]; es el [*maestro* | *instructor*] menos querido [*del departamento* | *por nosotros*], conocido [*principalmente* | *más*] en [*el* | *nuestro*] departamento por su [*tendencia* | *propensión*] a [*ridiculizar* | *avergonzar*] a los estudiantes lo bastante [*tontos* | *imprudentes*] como para hacer preguntas durante su clase.

[*Además* | *Por otra parte*], el profesor Wilson no es [*hábil* | *diligente*] para obtener financiamiento. Sus [*subvenciones* | *contratos*] han aportado una cantidad [*insustancial* | *insignificante*] de dinero [*al* | *a nuestro*] departamento. A menos que [*se recolecte dinero nuevo* | *se consigan fondos nuevos*] pronto tendremos que cancelar algunos programas esenciales, como su programa Estado 2000. Desgraciadamente, en estas [*condiciones* | *circunstancias*] no puedo recomendarlo de buena [*conciencia* | *fe*] a usted para [*la cátedra* | *un puesto permanente*].

Ahora Ellen prepara su computadora para calcular los 2^{32} compendios de mensaje para cada carta durante la noche. Con suerte, un compendio de la primera carta será igual a un compendio de la segunda. De no serlo puede agregar algunas opciones más e intentar de nuevo durante el fin de semana. Supongamos que encuentra una correspondencia. Llamemos a la carta “buena” *A* y a la “mala” *B*.

Ahora Ellen envía la carta *A* a Marilyn para su aprobación y mantiene en completo secreto la carta *B*, sin mostrársela a nadie. Marilyn, por supuesto, la aprueba, calcula su compendio de mensaje de 64 bits, firma el compendio y manda por correo electrónico el compendio firmado al rector Smith. Independientemente, Ellen envía la carta *B* al rector (no la carta *A*, que se suponía debía haber enviado).

Al recibir la carta y el compendio de mensaje firmado, el rector ejecuta el algoritmo de compendio de mensaje con la carta *B*, constata que coincide con lo que Marilyn le envió y despide a Tom. El rector no se da cuenta de que Ellen se las ingenió para generar dos cartas con el mismo compendio de mensaje y le envió una diferente de la que Marilyn vio y aprobó. (Desenlace opcio-

nal: Ellen le dice a Dick lo que hizo. Dick se horroriza y rompe con ella. Ellen se pone furiosa y confiesa su falta a Marilyn. Ésta llama al rector. Tom consigue la cátedra a fin de cuentas.) Con el MD5, el ataque de cumpleaños es difícil porque aun a una velocidad de mil millones de compendios por segundo, se requerirían más de 500 años para calcular los 2^{64} compendios de dos cartas con 64 variantes cada una, e incluso entonces no se garantiza una equivalencia. Por supuesto, con 5000 computadoras operando en paralelo, 500 años se convierten en 5 semanas. SHA-1 es mejor (porque es más largo).

8.5 ADMINISTRACIÓN DE CLAVES PÚBLICAS

La criptografía de clave pública hace posible que las personas que no comparten una clave común se comuniquen con seguridad. También posibilita firmar mensajes sin la presencia de un tercero confiable. Por último, los compendios de mensajes firmados hacen que verificar fácilmente la integridad de mensajes recibidos sea una realidad.

Sin embargo, hay un problema que hemos pasado por alto: si Alice y Bob no se conocen entre sí, ¿cómo obtiene cada uno la clave pública del otro para iniciar el proceso de comunicación? La solución más obvia —colocar su clave pública en su sitio Web— no funciona por la siguiente razón. Suponga que Alice quiere buscar la clave pública de Bob en el sitio Web de él. ¿Cómo lo hace? Comienza tecleando el URL de Bob. A continuación su navegador busca la dirección DNS de la página de inicio de Bob y le envía una solicitud *GET*, como se muestra en la figura 8-23. Desgraciadamente, Trudy intercepta la solicitud y responde con una página de inicio falsa, probablemente una copia de la de Bob, excepto por el reemplazo de la clave pública de Bob con la de Trudy. Cuando Alice encripta su primer mensaje con E_T, Trudy lo desencripta, lo lee, lo vuelve a encriptar con la clave pública de Bob y lo envía a éste, quien no tiene la menor idea de que Trudy está leyendo los mensajes que le llegan. Peor aún, Trudy puede modificar los mensajes antes de volverlos a encriptar para Bob. Claramente, se necesita un mecanismo para asegurar que las claves públicas puedan intercambiarse de manera segura.

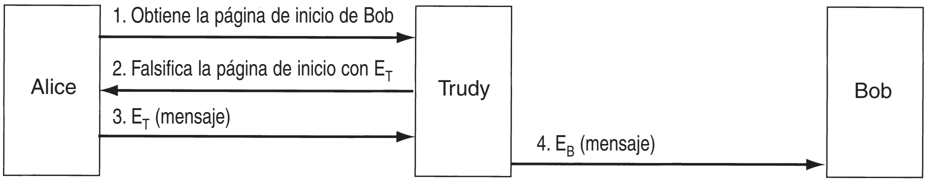


Figura 8-23. Una forma mediante la que Trudy puede subvertir la encriptación de clave pública.

8.5.1 Certificados

Como un primer intento para distribuir claves públicas de manera segura, podemos imaginar un centro de distribución de claves disponible en línea las 24 horas del día que proporciona claves públicas a petición. Uno de los muchos problemas con esta solución es que no es escalable, y el

centro de distribución de claves podría volverse rápidamente un cuello de botella. Además, si alguna vez fallara, la seguridad en Internet podría reducirse a nada.

Por estas razones, se ha desarrollado una solución diferente, una que no requiere que el centro de distribución esté en línea todo el tiempo. De hecho, ni siquiera tiene que estar en línea. En su lugar, lo que hace es certificar las claves públicas que pertenecen a las personas, empresas y otras organizaciones. Una organización que certifica claves públicas se conoce como **CA** (**autoridad de certificación**).

Como un ejemplo, suponga que Bob desea permitir que Alice y otras personas se comuniquen con él de manera segura. Él puede ir con la CA con su clave pública junto con su pasaporte o licencia de conducir para pedir su certificación. A continuación, la CA emite un certificado similar al que se muestra en la figura 8-24 y firma su *hash* SHA-1 con la clave privada de la CA. Bob paga la cuota de la CA y obtiene un disco flexible que contiene el certificado y su *hash* firmado.

```
Certifico que la clave pública
    19836A8B03030CF83737E3837837FC3s87092827262643FFA82710382828282A
pertenece a
    Robert John Smith
    12345 University Avenue
    Berkeley, CA 94702
    Birthday: July 4, 1958
    Email: bob@superdupernet.com
```

Hash SHA-1 del certificado anterior firmado con la clave pública de la CA

Figura 8-24. Un posible certificado y su *hash* firmado.

El trabajo fundamental de un certificado es enlazar una clave pública con el nombre de un personaje principal (individual, empresa, etcétera). Los certificados mismos no son secretos ni protegidos. Por ejemplo, Bob podría decidir colocar su nuevo certificado en su sitio Web, con un vínculo en la página de inicio que diga: Haga clic aquí para obtener mi certificado de clave pública. El clic resultante podría regresar el certificado y el bloque de la firma (el *hash* SHA-1 firmado del certificado).

Ahora vayamos nuevamente al escenario que se muestra en la figura 8-23. Cuando Trudy intercepta la solicitud que Alice realiza para obtener la página de inicio de Bob, ¿qué puede hacer Trudy? Puede poner su propio certificado y bloque de firma en la página falsificada, pero cuando Alice lea el certificado, verá inmediatamente que no está hablando con Bob porque el nombre de éste no se encuentra en dicho certificado. Trudy puede modificar sobre la marcha la página de inicio de Bob, reemplazando la clave privada de Bob con la suya. Sin embargo, cuando Alice ejecute el algoritmo SHA-1 en el certificado, obtendrá un *hash* que no corresponde con el que obtuvo cuando aplicó la clave privada bien conocida de la CA al bloque de firma. Puesto que Trudy no tiene la clave privada de la CA, no tiene forma de generar un bloque de firma que contenga el *hash* de la página Web modificada con su clave pública en él. De esta manera, Alice puede estar segura de que tiene la clave pública de Bob y no la de Trudy o la de alguien más. Y, como prometimos, este esquema no requiere que la CA esté en línea para la verificación, por lo tanto se elimina un cuello de botella potencial.

Mientras que la función estándar de un certificado es enlazar una clave pública a un personaje principal, un certificado también se puede utilizar para enlazar una clave pública a un **atributo.** Por ejemplo, un certificado podría decir: Esta clave pública pertenece a alguien mayor de 18 años. Podría utilizarse para probar que el dueño de la clave privada no es una persona menor de edad y, por lo tanto, se le permitió acceder material no apto para niños, entre otras cosas, pero sin revelar la identidad del dueño. Por lo general, la persona que tiene el certificado podría enviarlo al sitio Web, al personaje principal o al proceso que se preocupa por la edad. El sitio, el personaje principal o el proceso podrían generar a continuación un número aleatorio y encriptarlo con la clave pública del certificado. Si el dueño pudiera desencriptarlo y regresarlo, ésa sería una prueba de que el dueño tenía el atributo establecido en el certificado. De manera alternativa, el número aleatorio podría utilizarse para generar una clave de sesión para la conversación resultante.

Otro ejemplo en el que un certificado podría contener un atributo es un sistema distribuido orientado a objetos. Cada objeto normalmente tiene múltiples métodos. El dueño del objeto podría proporcionar a cada cliente un certificado que dé un mapa de bits de cuáles métodos puede invocar y que enlace dicho mapa de bits a una clave pública mediante un certificado firmado. Nuevamente, si el dueño del certificado puede probar la posesión de la clave privada correspondiente, se le permitirá realizar los métodos en el mapa de bits. Tiene la propiedad de que la identidad del dueño no necesita conocerse, lo cual es útil en las situaciones en las que la privacidad es importante.

8.5.2 X.509

Si todas las personas que desean algo firmado fueran a la CA con un tipo diferente de certificado, administrar todos los formatos diferentes pronto se volvería un problema. Para resolverlo se ha diseñado un estándar para certificados, el cual ha sido aprobado por la ITU. Dicho estándar se conoce como **X.509** y se utiliza ampliamente en Internet. Ha tenido tres versiones desde su estandarización inicial en 1988. Aquí analizaremos la versión V3.

El X.509 ha recibido una enorme influencia del mundo de OSI, y ha tomado prestadas algunas de sus peores características (por ejemplo, la asignación de nombres y la codificación). Sorprendentemente, la IETF estaba de acuerdo con el X.509, aunque en casi todas las demás áreas, desde direcciones de máquinas, protocolos de transporte hasta formatos de correo electrónico, la IETF por lo general ignoró a la OSI y trató de hacerlo bien. La versión IETF del X.509 se describe en el RFC 3280.

En esencia, el X.509 es una forma de describir certificados. Los campos principales en un certificado se listan en la figura 8-25. Las descripciones dadas ahí deben proporcionar una idea general de lo que hacen los campos. Para información adicional, por favor consulte el estándar mismo o el RFC 2459.

Por ejemplo, si Bob trabaja en el departamento de préstamos del Banco Monetario, su dirección X.500 podría ser:

```
/C=MX/O=BancoMonetario/OU=Prestamo/CN=Bob/
```

donde *C* corresponde al país, *O* a la organización, *OU* a la unidad organizacional y *CN* a un nombre común. Las CAs y otras entidades se nombran de forma similar. Un problema considerable con

Campo	Significado
Versión	Cuál versión del X.509
Número de serie	Este número junto con el nombre de la CA identifican de manera única el certificado
Algoritmo de firma	El algoritmo que se utilizó para firmar el certificado
Emisor	El nombre X.500 de la CA
Validez	Las fechas de inicio y final del periodo de validez
Nombre del sujeto	La entidad cuya clave se está certificando
Clave pública	La clave pública del sujeto y el ID del algoritmo usado para generarla
ID del emisor	Un ID opcional que identifica de manera única al emisor del certificado
ID del sujeto	Un ID opcional que identifica de manera única al sujeto del certificado
Extensiones	Se han definido muchas extensiones
Firma	La firma del certificado (firmada por la clave privada de la CA)

Figura 8-25. Los campos básicos de un certificado X.509.

los nombres X.500 es que si Alice está tratando de contactar a *bob@bancomonetario.com* y se le da un certificado con un nombre X.500, tal vez no sea obvio para ella que el certificado se refiera al Bob que ella busca. Por fortuna, a partir de la versión 3 se permiten los nombres DNS en lugar de los de X.500, por lo que este problema terminará por resolverse en algún momento.

Los certificados están codificados mediante la **ASN.1** (**Notación de Sintaxis Abstracta 1**) de la OSI, que puede considerarse como si fuera una estructura de C, pero con una notación peculiar y poco concisa. Es posible encontrar información sobre X.509 en (Ford y Baum, 2000).

8.5.3 Infraestructuras de clave pública

El hecho de que una sola CA emita todos los certificados del mundo obviamente no funciona. Podría derrumbarse por la carga y también podría ser un punto central de fallas. Una posible solución sería tener múltiples CAs que fueran ejecutadas por la misma organización y que utilizaran la misma clave privada para firmar los certificados. Si bien esto podría solucionar los problemas de carga y de fallas, introduciría un nuevo problema: la fuga de claves. Si hubiera docenas de servidores esparcidos por todo el mundo, todos con la misma clave privada de la CA, la probabilidad de que la clave privada fuera robada o filtrada se incrementaría de manera considerable. Puesto que la situación comprometida de esta clave arruinaría la infraestructura de la seguridad electrónica mundial, tener una sola CA central es muy peligroso.

Además, ¿qué organización podría operar la CA? Es difícil imaginar cualquier autoridad que podría ser aceptada mundialmente como legítima y digna de confianza. En algunos países las personas insistirían en que fuera el gobierno, mientras que en otros lo rechazarían.

Por estas razones, se ha desarrollado una forma diferente para certificar claves públicas. Tiene el nombre general **PKI** (**Infraestructura de Clave Pública**). En esta sección resumiremos cómo funciona, aunque ha habido diversas propuestas, por lo que los detalles podrían cambiar con el tiempo.

Una PKI tiene múltiples componentes, entre ellos usuarios, CAs, certificados y directorios. Lo que una PKI hace es proporcionar una forma para estructurar estos componentes y definir estándares para los diversos documentos y protocolos. Una forma particularmente simple de PKI es una jerarquía de CAs, como se muestra en la figura 8-26. En este ejemplo mostramos tres niveles, pero en la práctica podrían ser menos o más. La CA de nivel superior, la raíz, certifica a CAs de segundo nivel, a las que llamaremos **RAs** (**Autoridades Regionales**) debido a que podrían cubrir alguna región geográfica, como un país o un continente. Sin embargo, este término no es estándar; de hecho, ningún término es realmente estándar para los diversos niveles del árbol. Estas RAs, a su vez, certifican a los CAs reales, las cuales emiten los certificados X.509 a organizaciones e individuos. Cuando la raíz autoriza una nueva RA, genera un certificado X.509 donde indica que ha aprobado la RA, e incluye en él la nueva clave pública de la RA, la firma y se la proporciona a la RA. De manera similar, cuando una RA aprueba una CA, produce y firma un certificado que indica su aprobación y que contiene la clave pública de la CA.

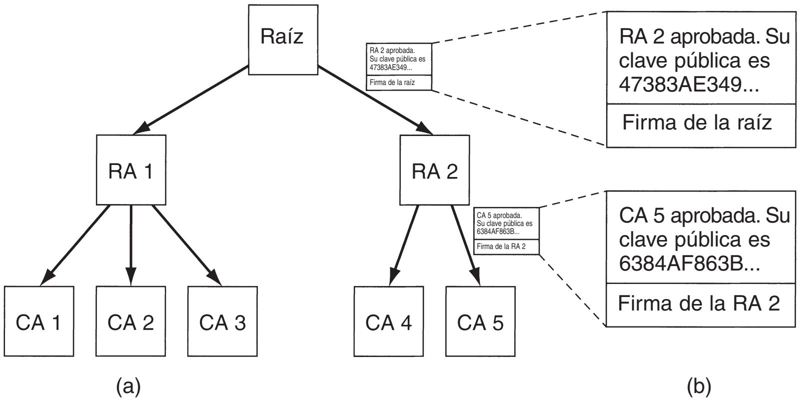


Figura 8-26. (a) Una PKI jerárquica. (b) Una cadena de certificados.

Nuestra PKI funciona como se muestra a continuación. Suponga que Alice necesita la clave pública de Bob para comunicarse con él, por lo que busca y encuentra un certificado que la contiene, firmado por la CA 5. Sin embargo, Alice nunca ha escuchado sobre la CA 5. Por lo que ella sabe, CA 5 podría ser la hija de 10 años de Bob. Puede ir con la CA 5 y decirle: Prueba tu autenticidad. La CA 5 responde con el certificado que obtuvo de la RA 2, el cual contiene la clave pública de la CA 5. Una vez que tiene la clave pública de la CA 5, Alice puede verificar que el certificado de Bob realmente fue firmado por la CA 5 y que, por lo tanto, es legal.

Para asegurarse de que la RA 2 no sea el hijo de 12 años de Bob, el siguiente paso es que Alice pida a la RA 2 que pruebe su autenticidad. La respuesta a su consulta es un certificado firmado por la raíz que contiene la clave pública de la RA 2. Ahora Alice está segura de que tiene la clave pública de Bob.

Pero, ¿cómo averigua Alice la clave pública de la raíz? Magia. Se supone que todos conocen la clave pública de la raíz. Por ejemplo, su navegador podría haberse enviado con la clave pública de la raíz integrada en él.

Bob es una persona amigable y no desea causar a Alice tanto trabajo. Él sabe que ella va a tener que verificar la CA 5 y la RA 2, por lo que para ahorrarle algunos problemas, recolecta los dos certificados necesarios y se los proporciona junto con el de él. Ahora puede utilizar el conocimiento que tiene de la clave pública de la raíz para verificar el certificado de nivel superior y la clave pública contenida ahí para verificar la segunda. De esta manera, Alice no necesita contactar a nadie para realizar la verificación. Debido a que todos los certificados están firmados, Alice puede detectar con facilidad cualquier intento de alterar el contenido. Una cadena de certificados que va de esta forma a la raíz algunas veces se conoce como **cadena de confianza** o **ruta de certificación.** La técnica se utiliza ampliamente en la práctica.

Por supuesto, aún tenemos el problema de quién va a ejecutar la raíz. La solución no es tener una sola raíz, sino tener muchas, cada una con sus propias RAs y CAs. De hecho, los navegadores modernos vienen precargados con claves públicas para aproximadamente 100 raíces, algunas veces llamadas **anclas de confianza.** De esta forma, es posible evitar tener una sola autoridad mundial confiable.

Pero ahora queda el problema de en qué forma el fabricante del navegador decide cuáles anclas de confianza propuestas son confiables y cuáles no. Queda a criterio del usuario confiar en el fabricante del navegador para elegir las mejores opciones y no simplemente aprobar todas las anclas de confianza deseando pagar sus cuotas de inclusión. La mayoría de los navegadores permiten que los usuarios inspeccionen las claves de la raíz (por lo general en la forma de certificados firmados por la raíz) y eliminen las que parezcan sospechosas.

Directorios

Otro problema de cualquier PKI es en dónde están almacenados los certificados (y sus cadenas hacia un ancla de confianza conocida). Una posibilidad es hacer que cada usuario almacene sus propios certificados. Si bien esto es seguro (es decir, no hay forma de que los usuarios falsifiquen certificados firmados sin que esto se detecte), también es inconveniente. Una alternativa que se ha propuesto es utilizar DNS como un directorio de certificados. Antes de contactar a Bob, Alice probablemente tiene que buscar la dirección IP de Bob mediante DNS, entonces, ¿por qué no hacer que DNS devuelva toda la cadena de certificados de Bob junto con su dirección IP?

Algunas personas piensan que ésta es una forma de proceder, pero tal vez otras prefieren dedicar servidores de directorio cuyo único trabajo sea manejar los certificados X.509. Tales directorios podrían proporcionar servicios de búsqueda utilizando propiedades de los nombres X.500. Por ejemplo, en teoría un servicio de directorio como éste podría contestar una consulta como: “Dame una lista de todas las personas que tengan el nombre Alice y que trabajen en los departamentos de ventas en cualquier lugar de Estados Unidos o Canadá”. LDAP podría ser un candidato para almacenar esta información.

Revocación

El mundo real también está lleno de certificados, como los pasaportes y las licencias de conducir. Algunas veces estos certificados pueden revocarse, por ejemplo, las licencias de conducir pueden revocarse por conducir en estado de ebriedad y por otros delitos de manejo. En el mundo digital ocurre el mismo problema: el otorgante de un certificado podría decidir revocarlo porque la persona u organización que lo posee ha abusado de alguna manera. También puede revocarse si la clave privada del sujeto se ha expuesto o, peor aún, si la clave privada de la CA está en peligro. Por lo tanto, una PKI necesita tratar el problema de la revocación.

Un primer paso en esta dirección es hacer que cada CA emita periódicamente una **CRL** (**lista de revocación de certificados**) que proporcione los números seriales de todos los certificados que ha revocado. Puesto que los certificados contienen fechas de vencimiento, la CRL sólo necesita contener los números seriales de los certificados que no han expirado. Una vez que pasa la fecha de vencimiento de un certificado, éste se invalida de manera automática, por lo que no hay necesidad de hacer una distinción entre los certificados que han expirado y los que fueron revocados. Ninguno de esos tipos de certificados puede utilizarse.

Desgraciadamente, introducir CRLs significa que un usuario que está próximo a utilizar un certificado debe adquirir la CRL para ver si su certificado ha sido revocado. Si es así, dicho certificado no debe utilizarse. Sin embargo, si el certificado no está en la lista, pudo haber sido revocado justo después de que se publicó la lista. Por lo tanto, la única manera de estar seguro realmente es preguntar a la CA. Y la siguiente vez que se utilice ese mismo certificado, se le tiene que preguntar nuevamente a la CA, puesto que dicho certificado pudo haber sido revocado segundos antes.

Otra complicación es que un certificado revocado puede reinstalarse nuevamente, por ejemplo, si fue revocado por falta de pago, pero ahora se ha puesto al corriente. Tener que tratar con la revocación (y, posiblemente, con la reinstalación) elimina una de la mejores propiedades de los certificados, principalmente, que pueden utilizarse sin tener que contactar a una CA.

¿Dónde deben almacenarse las CRLs? Un buen lugar sería el mismo en el que se almacenan los certificados. Una estrategia es que una CA quite de manera activa y periódica CRLs y hacer que los directorios las procesen con sólo eliminar los certificados revocados. Si no se utilizan directorios para almacenar certificados, las CRLs pueden almacenarse en caché en varios lugares convenientes alrededor de la red. Puesto que una CRL es por sí misma un documento firmado, si se altera, esa alteración puede detectarse con facilidad.

Si los certificados tienen tiempos de vida largos, las CRLs también los tendrán. Por ejemplo, si las tarjetas de crédito son válidas durante cinco años, el número de revocaciones pendientes será mucho más grande que si se emitieran nuevas tarjetas cada tres meses. Una forma estándar para tratar con CRLs grandes es emitir una lista maestra ocasionalmente, pero emitir actualizaciones con más frecuencia. Hacer esto reduce el ancho de banda necesario para distribuir las CRLs.

8.6 SEGURIDAD EN LA COMUNICACIÓN

Hemos terminado nuestro estudio de las herramientas en cuestión. Ya cubrimos la mayoría de las técnicas y protocolos importantes. El resto del capítulo trata de cómo se utilizan estas técnicas en la práctica para proporcionar seguridad de red, más algunas reflexiones sobre los aspectos sociales de la seguridad, al final del capítulo.

En las siguientes cuatro secciones veremos la seguridad en la comunicación, es decir, cómo obtener los bits de manera secreta y sin modificación desde el origen hasta el destino y cómo mantener fuera a los bits no deseados. Éstos no son de ningún modo los únicos aspectos de seguridad en las redes, pero ciertamente están entre los más importantes, lo que hace de éste un buen lugar para comenzar.

8.6.1 IPsec

La IETF ha sabido por años que hay una falta de seguridad en Internet. Agregarla no era fácil pues surgió una controversia sobre dónde colocarla. La mayoría de los expertos en seguridad creían que para estar realmente seguro, el cifrado y las verificaciones de integridad tenían que llevarse a cabo de extremo a extremo (es decir, en la capa de aplicación). De tal manera, el proceso de origen encripta y/o protege la integridad de los datos y los envía al proceso de destino en donde se desencriptan y/o verifican. Por lo tanto, cualquier alteración hecha en medio de estos dos procesos, o en cualquier sistema operativo, puede detectarse. El problema con este enfoque es que requiere cambiar todas las aplicaciones para que estén conscientes de la seguridad. Desde esta perspectiva, el siguiente mejor enfoque es colocar el cifrado en la capa de transporte o en una nueva capa entre la capa de aplicación y la de transporte, con lo que se conserva el enfoque de extremo a extremo pero no requiere que se cambien las aplicaciones.

La perspectiva opuesta es que los usuarios no entiendan la seguridad y no sean capaces de utilizarla correctamente, así como que nadie desee modificar los programas existentes de ninguna forma, por lo que la capa de red debe autentificar y/o cifrar paquetes sin que los usuarios estén involucrados. Después de años de batallas encarnizadas, esta perspectiva ganó soporte suficiente para que se definiera un estándar de seguridad de capa de red. El argumento fue en parte que tener cifrado de la capa de red no evitaba que los usuarios conscientes de la seguridad la aplicaran correctamente y que ayudaba hasta cierto punto a los usuarios no conscientes de ella.

El resultado de esta guerra fue un diseño llamado **IPsec** (**Seguridad IP**), que se describe en los RFCs 2401, 2402 y 2406, entre otros. No todos los usuarios desean cifrado (pues éste es costoso computacionalmente). En lugar de hacerlo opcional, se decidió requerir cifrado todo el tiempo pero permitir el uso de un algoritmo nulo. Éste se describe y alaba por su simplicidad, facilidad de implementación y gran velocidad en el RFC 2410.

El diseño IPsec completo es una estructura para servicios, algoritmos y granularidades múltiples. La razón para los servicios múltiples es que no todas las personas quieren pagar el precio por tener todos los servicios todo el tiempo, por lo que los servicios están disponibles a la carta. Los servicios principales son confidencialidad, integridad de datos y protección contra ataques de

repetición (un intruso repite una conversación). Todos estos se basan en criptografía simétrica debido a que el alto rendimiento es crucial.

La razón de tener múltiples algoritmos es que un algoritmo que ahora se piensa es seguro puede ser violado en el futuro. Al hacer independiente al algoritmo IPsec, la estructura puede sobrevivir incluso si algún algoritmo particular es violado posteriormente.

La razón de tener múltiples granularidades es para hacer posible la protección de una sola conexión TCP, todo el tráfico entre un par de *hosts* o todo el tráfico entre un par de enrutadores seguros, entre otras posibilidades.

Un aspecto ligeramente sorprendente de IPsec es que aunque se encuentra en la capa IP, es orientado a la conexión. En la actualidad, eso no es tan sorprendente porque para tener seguridad, se debe establecer y utilizar una clave por algún periodo —en esencia, un tipo de conexión. Además, las conexiones amortizan los costos de configuración sobre muchos paquetes. Una "conexión" en el contexto de IPsec se conoce como **SA** (**asociación de seguridad).** Una SA es una conexión símplex entre dos puntos finales y tiene un identificador de seguridad asociado con ella. Si se necesita tráfico seguro en ambas direcciones, se requieren dos asociaciones de seguridad. Los identificadores de seguridad se transportan en paquetes que viajan en estas conexiones seguras y se utilizan para buscar claves y otra información relevante cuando llega un paquete seguro.

Técnicamente, IPsec tiene dos partes principales. La primera describe dos encabezados nuevos que pueden agregarse a paquetes para transportar el identificador de seguridad, datos de control de integridad, entre otra información. La otra parte, **ISAKMP** (**Asociación para Seguridad en Internet y Protocolo de Administración de Claves),** tiene que ver con el establecimiento de claves. No trataremos con mayor detalle a ISAKMP porque (1) es extremadamente complejo y (2) su protocolo principal, **IKE** (**Intercambio de Claves de Internet),** está plagado de fallas y necesita ser reemplazado (Perlman y Kaufman, 2000).

IPsec puede utilizarse en cualquiera de dos modos. En el **modo de transporte,** el encabezado IPsec se inserta justo después del encabezado IP. El campo *Protocolo* del encabezado IP se cambia para indicar que un encabezado IPsec sigue al encabezado IP normal (antes del encabezado TCP). El encabezado IPsec contiene información de seguridad, principalmente el identificador SA, un nuevo número de secuencia y tal vez una verificación de integridad del campo de carga.

En el **modo de túnel,** todo el paquete IP, encabezado y demás, se encapsula en el cuerpo de un paquete IP nuevo con un encabezado IP completamente nuevo. El modo de túnel es útil cuando un túnel termina en una ubicación que no sea el destino final. En algunos casos, el final del túnel es una máquina de puerta de enlace de seguridad, por ejemplo, un *firewall* de una empresa. En este modo, el *firewall* encapsula y desencapsula paquetes conforme pasan a través del *firewall*. Al terminar el túnel en esta máquina segura, las máquinas en la LAN de la empresa no tienen que estar consciente de IPsec. Sólo el *firewall* tiene que saber sobre él.

El modo de túnel también es útil cuando se agrega un conjunto de conexiones TCP y se maneja como un solo flujo cifrado porque así se evita que in intruso vea quién está enviando cuántos paquetes a quién. Algunas veces el simple hecho de saber cuánto tráfico está pasando y hacia dónde se dirige es información valiosa. Por ejemplo, si durante una crisis militar, la cantidad de tráfico que fluye entre el Pentágono y la Casa Blanca se reduce de manera significativa, pero la cantidad de tráfico entre el Pentágono y alguna instalación militar oculta entre las Montañas Rocosas de

Colorado se incrementa en la misma cantidad, un intruso podría ser capaz de deducir alguna información útil a partir de estos datos. El estudio de los patrones de flujo de paquetes, aunque estén cifrados, se conoce como **análisis de tráfico.** El modo de túnel proporciona una forma de frustrarlo hasta cierto punto. La desventaja del modo de túnel es que agrega un encabezado IP extra, por lo que se incrementa el tamaño del paquete en forma sustancial. En contraste, el modo de transporte no afecta tanto el tamaño del paquete.

El primer nuevo encabezado es **AH** (**encabezado de autenticación**)**.** Proporciona verificación de integridad y seguridad antirrepetición, pero no la confidencialidad (es decir, no cifrado de datos). El uso de AH en el modo de transporte se ilustra en la figura 8-27. En el IPv4 se coloca entre el encabezado IP (incluyendo cualquier opción) y el TCP. En IPv6 es sólo otro encabezado de extensión y se trata como tal. De hecho, el formato está cerca del de un encabezado de extensión IPv6 estándar. La carga útil tal vez tenga que rellenarse a alguna longitud en particular para el algoritmo de autenticación, como se muestra.

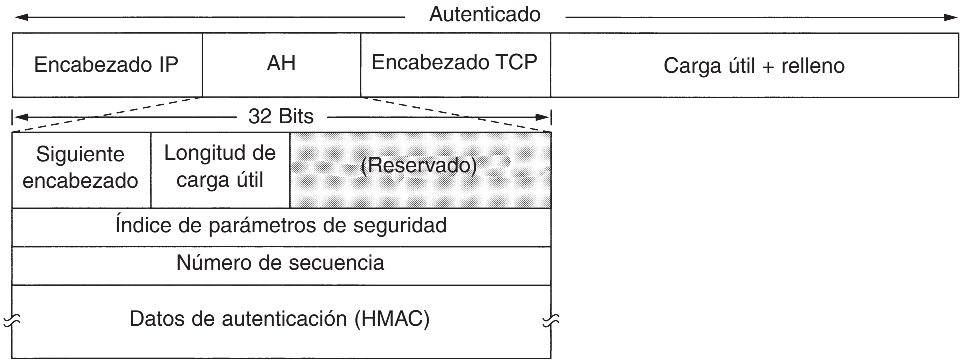


Figura 8-27. El encabezado de autenticación IPsec en el modo de transporte para IPv4.

Examinemos el encabezado AH. El campo *Siguiente encabezado* se utiliza para almacenar el valor anterior que tenía el campo *Protocolo* IP antes de que se reemplazara con 51 para indicar que seguía un encabezado AH. En muchos casos, el código para TCP (6) irá aquí. La *Longitud de carga útil* es el número de palabras de 32 bits en el encabezado AH menos 2.

El *Índice de parámetros de seguridad* es el indicador de conexión. Es insertado por el emisor para indicar un registro en particular en la base de datos del receptor. Este registro contiene la clave compartida utilizada en esta conexión y otra información sobre dicha conexión. Si este protocolo hubiera sido inventado por la ITU en lugar del IETF, este campo se hubiera llamado *Número de circuitos virtuales*.

El campo *Número de secuencia* se utiliza para numerar todos los paquetes enviados en una SA. Cada paquete obtiene un número único, incluso las retransmisiones. En otras palabras, la retransmisión de un paquete obtiene un número diferente aquí que el original (aunque su número de secuencia TCP sea el mismo). El propósito de este campo es detectar ataques de repetición. Tal vez estos números de secuencia no se ajusten. Si todos los 2^{32} se agotan, debe establecerse una nueva SA para continuar la comunicación.

Por último, veamos el campo *Datos de autenticación*, que es de longitud variable y contiene la firma digital de la carga útil. Cuando se establece la SA, los dos lados negocian cuál algoritmo de firmas van a utilizar. Por lo general, aquí no se utiliza la criptografía de clave pública porque los paquetes se deben procesar extremadamente rápido y todos los algoritmos de clave pública conocidos son muy lentos. Puesto que IPsec se basa en la criptografía de clave simétrica y el emisor y el receptor negocian una clave compartida antes de establecer una SA, dicha clave compartida se utiliza en el cálculo de la firma. Una forma simple es calcular el *hash* sobre el paquete más la clave compartida. Por supuesto, ésta no se transmite. Un esquema como éste se conoce como **HMAC** (**Código de Autenticación de Mensajes basado en *Hash*).** Es más rápido realizar el cálculo que primero ejecutar el SHA-1 y luego el RSA sobre el resultado.

El encabezado AH no permite la encriptación de los datos, por lo que es útil principalmente cuando la verificación de la integridad es necesaria pero la confidencialidad no lo es. Una característica de AH que vale la pena es que la verificación de integridad abarca algunos de los campos en el encabezado IP, principalmente aquellos que no cambian conforme el paquete se mueve de enrutador a enrutador. El campo *Tiempo de vida* cambia en cada salto, por ejemplo, de manera que no se puede incluir en la verificación de integridad. Sin embargo, la dirección IP de origen se incluye en la verificación, lo que hace imposible que un intruso falsifique el origen de un paquete.

El encabezado IPsec alternativo es **ESP** (**Carga Útil de Encapsulamiento de Seguridad).** Su uso para modo de transporte y para modo de túnel se muestra en la figura 8-28.

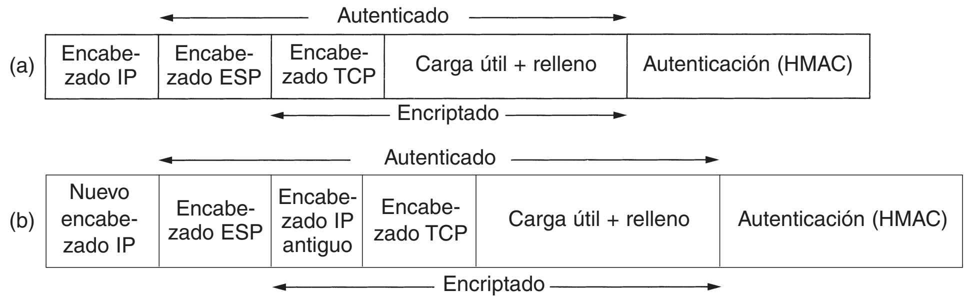


Figura 8-28. (a) ESP en modo de transporte. (b) ESP en modo de túnel.

El encabezado ESP consiste en dos palabras de 32 bits. Éstas son los campos *Índice de parámetros de seguridad* y *Número de secuencia* que vimos en AH. Una tercera palabra que generalmente va después de ellos (pero que técnicamente no es parte del encabezado) es el *Vector de inicialización* utilizado para la encriptación de datos, a menos que se utilice la encriptación nula, en cuyo caso se omite.

ESP también incluye verificaciones de integridad HMAC, al igual que AH, pero en lugar de incluirse en el encabezado, van después de la carga útil, como se muestra en la figura 8-28. Colocar el HMAC al final tiene una ventaja en una implementación de hardware. El HMAC se puede calcular conforme los bits se transmiten por la interfaz de red y agregarse al final. Ésta es la razón por la cual Ethernet y otras LANs tienen sus CRCs en un terminador, en lugar de en un encabezado. Con AH, el paquete tiene que almacenarse en el búfer y la firma tiene que calcularse antes

de que se envíe el paquete, con lo que se reduce de manera potencial el número de paquetes/seg que pueden enviarse.

Puesto que ESP puede hacer lo mismo que AH y más, y debido a que es más eficiente para iniciar, surge la pregunta: ¿Por qué molestarse en tener a AH? La respuesta es principalmente histórica. En un principio, AH manejaba sólo integridad y ESP manejaba sólo confidencialidad. Más tarde, la integridad se agregó a ESP, pero las personas que diseñaron AH no querían dejarlo morir después de todo el trabajo que habían realizado. Sin embargo, su único argumento es que AH verifica parte del encabezado IP, lo cual ESP no hace, pero es un argumento débil. Otro argumento débil es que un producto que soporta AH y no ESP podría tener menos problemas para obtener una licencia de exportación porque no puede realizar encriptación. Es probable que AH sea desplazado en el futuro.

8.6.2 *Firewalls*

La capacidad de conectar una computadora, en cualquier lugar, con cualquier computadora, de cualquier lugar, es una ventaja a medias. Para los usuarios domésticos, navegar en Internet significa mucha diversión. Para los gerentes de seguridad empresarial, es una pesadilla. Muchas empresas tienen en línea grandes cantidades de información confidencial —secretos de comercio, planes de desarrollo de productos, estrategias de marketing, análisis financieros, etcétera. Si esta información cae en manos de un competidor podría tener graves consecuencias.

Además del peligro de la fuga de información, también existe el peligro de la infiltración de información. En particular, virus, gusanos y otras pestes digitales pueden abrir brechas de seguridad, destruir datos valiosos y hacer que los administradores pierdan mucho tiempo tratando de arreglar el daño que hayan hecho. Por lo general, son traídos por empleados descuidados que desean ejecutar algún nuevo juego ingenioso.

En consecuencia, se necesitan mecanismos para mantener adentro a los bits "buenos" y afuera a los bits "malos". Un método es utilizar IPsec. Este método protege a los datos en tránsito entre los sitios seguros. Sin embargo, IPsec no hace nada para mantener afuera de la LAN de la compañía a las pestes digitales y a los intrusos. Para saber cómo alcanzar ese objetivo, necesitamos ver los *firewalls*.

Los ***firewalls*** (servidores de seguridad) son simplemente una adaptación moderna de la vieja estrategia medieval de seguridad: excavar un foso defensivo profundo alrededor de su castillo. Este diseño obligaba a que todos los que entraran o salieran del castillo pasaran a través de un puente levadizo, en donde los encargados de la E/S los podían inspeccionar. En las redes es posible el mismo truco: una compañía puede tener muchas LANs conectadas de formas arbitrarias, pero se obliga a que todo el tráfico desde o hacia la compañía pase a través de un puente levadizo electrónico (*firewall*), como se muestra en la figura 8-29.

En esta configuración el *firewall* tiene dos componentes: dos enrutadores que realizan filtrado de paquetes y una puerta de enlace de aplicación. También existen configuraciones más simples, pero la ventaja de este diseño es que cada paquete debe transitar a través de dos filtros y una puerta de enlace de aplicación para entrar o salir. No existe otra ruta. Es evidente que quienes piensen que un punto de verificación de seguridad es suficiente, no han hecho vuelos internacionales recientemente en alguna aerolínea.

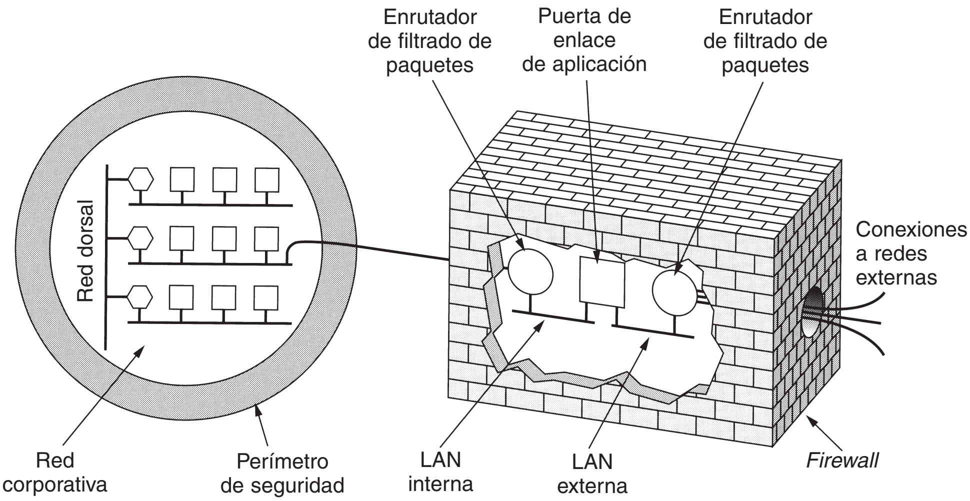


Figura 8-29. Un firewall que consiste en dos filtros de paquetes y en una puerta de enlace de aplicación.

Cada **filtro de paquete** es un enrutador estándar equipado con alguna funcionalidad extra. Ésta permite inspeccionar cada paquete entrante o saliente. Los paquetes que cumplan con algún criterio se reenvían de manera normal. Los que fallen la prueba se descartan.

En la figura 8-29 probablemente el filtro de paquete de la LAN interna verifica los paquetes salientes y el que está en la LAN externa verifica los paquetes entrantes. Los paquetes que cruzan la primera barrera pasan a la puerta de enlace de aplicación, donde se vuelven a examinar. El objetivo de colocar los dos filtros de paquetes en LANs diferentes es asegurar que ningún paquete entre o salga sin pasar a través de la puerta de enlace de aplicación: no hay otra ruta.

Por lo general, los filtros de paquetes son manejados por tablas configuradas por el administrador del sistema. Dichas tablas listan orígenes y destinos aceptables, orígenes y destinos bloqueados, y reglas predeterminadas sobre lo que se debe hacer con los paquetes que van o vienen de otras máquinas.

En el caso común de una configuración TCP/IP, un origen o un destino consiste en una dirección IP y un puerto. Los puertos indican qué servicio se desea. Por ejemplo, el puerto TCP 23 es para telnet, el puerto TCP 79 es para directorio y el puerto TCP 119 es para las noticias USENET. Una compañía podría bloquear los paquetes entrantes de todas las direcciones IP combinadas con uno de estos puertos. De esta forma, nadie afuera de la compañía puede iniciar una sesión a través de telnet o buscar personas mediante el demonio de directorio. Además, la compañía podría estar en contra de que sus empleados desperdicien todo el día leyendo noticias USENET.

Bloquear paquetes salientes es un poco difícil pues aunque la mayoría de los sitios se apegan a las convenciones estándar de numeración de puertos, no están obligados a hacerlo. Además, para algunos servicios importantes, como FTP (Protocolo de Transferencia de Archivos), los números de puerto se asignan de manera dinámica. Asimismo, aunque el bloqueo de conexiones TCP es difícil, el de paquetes UDP lo es aún más debido a que se sabe muy poco con anticipación sobre lo

que harán. Muchos filtros de paquetes están configurados para simplemente prohibir el tráfico de paquetes UDP.

La segunda mitad del *firewall* es la **puerta de enlace de aplicación.** En lugar de sólo buscar los paquetes simples, la puerta de enlace opera a nivel de aplicación. Por ejemplo, es posible configurar una puerta de enlace de correo para examinar cada mensaje que sale o entra. Para cada uno, la puerta de enlace decide si transmitir o descartar el mensaje con base en los campos de encabezado, el tamaño del mensaje o incluso en el contenido (por ejemplo, en una instalación militar, la presencia de palabras como “nuclear” o “bomba” podría causar que se tomara alguna acción especial).

Las instalaciones son libres de configurar una o más puertas de enlace de aplicación para aplicaciones específicas, aunque no es poco común que organizaciones desconfiadas permitan correo entrante y saliente, e incluso el acceso a World Wide Web, pero que consideren todo lo demás muy peligroso. Combinado con la encriptación y el filtrado de paquetes, este arreglo ofrece una cantidad limitada de seguridad con el costo de algunas inconveniencias.

Incluso si el *firewall* está configurado perfectamente, aún existirá una gran cantidad de problemas. Por ejemplo, si un *firewall* está configurado para permitir sólo paquetes entrantes de redes específicas (por ejemplo, de otras instalaciones de la compañía), un intruso afuera del *firewall* puede introducir direcciones de origen falsas para evadir esta verificación. Si un miembro interno desea enviar documentos secretos, puede encriptarlos o incluso fotografiarlos y enviar las fotos como archivos JPEG, los cuales pueden evadir cualquier filtro de palabras. Y no hemos analizado el hecho de que el 70% de todos los ataques provienen del lado interno del *firewall*, por ejemplo, de empleados descontentos (Schneier, 2000).

Además, hay otra clase de ataques que los *firewalls* no pueden manejar. La idea básica de un *firewall* es evitar que entren intrusos y que salga información secreta. Desgraciadamente, hay personas que no tienen nada mejor que hacer que tratar de perjudicar ciertos sitios. Esto lo hacen enviando grandes cantidades de paquetes legítimos al destino, hasta que el sitio se caiga debido a la carga. Por ejemplo, para derribar un sitio Web, un intruso puede enviar un paquete TCP *SYN* para establecer una conexión. A continuación el sitio asignará una ranura de tabla para la conexión y enviará como respuesta un paquete *SYN* + *ACK*. Si el intruso no responde, la ranura de tabla se conservará durante algunos segundos hasta que expire. Si el intruso envía miles de solicitudes de conexión, todas las ranuras de tabla se llenarán y no podrá establecerse ninguna conexión legítima. Los ataques en los que el objetivo del intruso es bloquear el destino en lugar de robar datos se conocen como ataques **DoS** (**negación de servicio**)**.** Por lo general, los paquetes de solicitud tienen direcciones falsas de origen por lo que el intruso no puede ser rastreado con facilidad.

Una variante aún peor es aquella en la que el intruso ha entrado en cientos de computadoras en cualquier parte del mundo, y después ordena a todas ellas que ataquen al mismo objetivo al mismo tiempo. Este método no sólo incrementa el poder del intruso, también reduce la probabilidad de su detección, debido a que los paquetes provienen de una gran cantidad de máquinas que pertenecen a usuarios ingenuos. Un ataque de este tipo se conoce como ataque **DDoS** (**negación de servicio distribuida**)**.** Es difícil defenderse de un ataque como éste. Incluso si la máquina atacada puede reconocer rápidamente una solicitud falsa, le toma algún tiempo procesar y descartar la

solicitud, y si llegan suficientes solicitudes por segundo, la CPU pasará todo su tiempo ocupándose de ellas.

8.6.3 Redes privadas virtuales

Muchas compañías tienen oficinas e instalaciones esparcidas en muchas ciudades, algunas veces en múltiples países. En el pasado, antes de que existieran las redes de datos públicas, era común que algunas compañías alquilaran líneas a las compañías telefónicas entre todas o entre sólo algunas ubicaciones. Algunas compañías aún hacen esto. Una red constituida por computadoras de compañías y líneas telefónicas alquiladas se conoce como **red privada.** En la figura 8-30(a) se muestra una red privada de ejemplo que conecta tres ubicaciones.

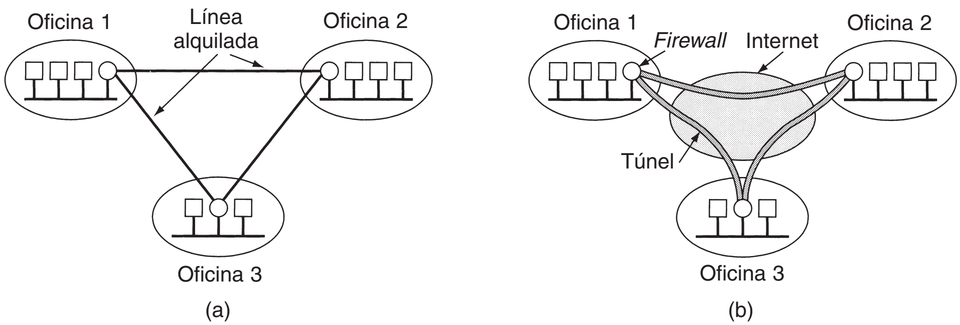


Figura 8-30. (a) Una red privada con línea alquilada. (b) Una red privada virtual.

Las redes privadas funcionan bien y son muy seguras. Si las únicas líneas disponibles son las alquiladas, el tráfico no puede fugarse de las ubicaciones de la compañía y los intrusos tienen que intervenir físicamente las líneas para infiltrarse, lo cual no es fácil de hacer. El problema con las redes privadas es que alquilar una sola línea T1 cuesta miles de dólares mensuales y las líneas T3 son muchas veces más costosas. Cuando aparecieron las redes de datos públicas y, más tarde, Internet, muchas compañías quisieron trasladar su tráfico de datos (y, posiblemente, de voz) a la red pública, aunque sin renunciar a la seguridad de la red privada.

Esta demanda pronto llevó a la invención de las **VPNs** (**redes privadas virtuales**), que son redes superpuestas sobre redes públicas pero con muchas propiedades de las redes privadas. Se llaman "virtuales" porque son sólo una ilusión, al igual que los circuitos virtuales no son circuitos reales ni la memoria virtual es memoria real.

Aunque las VPNs pueden implementarse encima de ATM (o de Frame Relay), un método cada vez más popular es construir VPNs directamente sobre Internet. Un diseño común es equipar cada oficina con un *firewall* y crear túneles a través de Internet entre todos los pares de oficinas,

como se ilustra en la figura 8-30(b). Si IPsec se utilizara para el proceso de entunelamiento, entonces sería posible agregar todo el tráfico entre cualquiera de los dos pares de oficinas en una sola SA encriptada y autenticada, con lo que se proporcionaría control de integridad, confidencialidad e incluso inmunidad considerable al análisis de tráfico.

Cuando se inicia el sistema, cada par de *firewalls* tiene que negociar los parámetros de su SA, incluyendo los servicios, modos, algoritmos y claves. Muchos *firewalls* tienen capacidades VPN integradas, aunque algunos enrutadores ordinarios también pueden hacer esto. Pero debido a que los *firewalls* están principalmente en el negocio de la seguridad, es natural que los túneles empiecen y terminen en los *firewalls*, estableciendo una clara separación entre la compañía e Internet. Por lo tanto, los *firewalls*, las VPNs e IPsec con ESP en modo de túnel son una combinación natural y se utilizan ampliamente en la práctica.

Una vez que se han establecido las SAs, el tráfico puede comenzar a fluir. Para un enrutador en Internet, un paquete que viaja a través de un túnel VPN es sólo un paquete ordinario. Lo único extraño es la presencia del encabezado IPsec después del encabezado IP, pero debido a que estos encabezados adicionales no tienen efecto en el proceso de reenvío, los enrutadores no se preocupan por ellos.

Una ventaja principal de organizar de esta forma una VPN es que es completamente transparente para todo el software de usuario. Los *firewalls* configuran y manejan las SAs. La única persona que está consciente de esta configuración es el administrador del sistema, quien tiene que configurar y manejar los *firewalls*. Para todos los demás, es como tener nuevamente una red privada mediante una línea alquilada. Para mayor información sobre las VPNs, vea (Brown, 1999, e Izzo, 2000).

8.6.4 Seguridad inalámbrica

Diseñar un sistema que sea lógica y completamente seguro mediante VPNs y *firewalls* es muy fácil, pero eso, en la práctica, puede fallar. Esta situación puede darse si algunas de las máquinas son inalámbricas y utilizan comunicación de radio, que pase justo encima del *firewall* en ambas direcciones. El rango de las redes 802.11 con frecuencia es de algunos cientos de metros, por lo que cualquiera que desee espiar una compañía puede simplemente introducirse en el estacionamiento para empleados por la mañana, dejar una computadora portátil habilitada para 802.11 en el automóvil para que grabe todo lo que oiga. Al anochecer, el disco duro estará repleto de datos valiosos. En teoría, se supone que esta fuga no debería suceder. Pero, en teoría, las personas tampoco deberían robar bancos.

La mayor parte del problema de seguridad puede remontarse a los fabricantes de las estaciones base inalámbricas (puntos de acceso), quienes tratan de hacer que sus productos sean amigables para el usuario. Por lo general, si el usuario saca el dispositivo de la caja y lo conecta en el enchufe de la energía eléctrica, comienza a operar inmediatamente —casi siempre sin seguridad, revelando secretos a quienes estén dentro del rango de radio. Si a continuación se conecta a una Ethernet, de pronto todo el tráfico de ésta aparecerá también en el estacionamiento. La tecnología

inalámbrica es el sueño de todo espía: datos gratuitos sin tener que hacer nada. Por lo tanto, sobra decir que la seguridad es mucho más importante para los sistemas inalámbricos que para los cableados. En esta sección veremos algunas formas en que las redes inalámbricas manejan la seguridad. Es posible encontrar información adicional en (Nichols y Lekkas, 2002).

Seguridad del 802.11

El estándar 802.11 establece un protocolo de seguridad en el nivel de capa de enlace de datos llamado **WEP** (**Privacidad Inalámbrica Equivalente**), diseñado para que la seguridad de una LAN inalámbrica sea tan buena como la de una LAN cableada. Puesto que lo predeterminado para las LANs alámbricas no es la seguridad, este objetivo es fácil de alcanzar, y WEP lo consigue, como veremos más adelante.

Cuando se habilita la seguridad para el estándar 802.11, cada estación tiene una clave secreta que comparte con la estación base. La forma en que se distribuyen las claves no se especifica en el estándar. Éstas sólo pueden ser precargadas por el fabricante. Pueden intercambiarse por adelantado a través de la red alámbrica. Por último, la estación base o la máquina del usuario pueden tomar una clave aleatoria y enviársela al otro por aire encriptada con la clave pública del otro. Una vez establecidas, por lo general las claves permanecen estables por meses o años.

La encriptación WEP utiliza un cifrado de flujo con base en el algoritmo RC4. Éste fue diseñado por Ronald Rivest y se mantuvo en secreto hasta que fue filtrado y se publicó en Internet en 1994. Como señalamos anteriormente, es casi imposible mantener en secreto los algoritmos, incluso cuando el objetivo es proteger la propiedad intelectual (como fue en este caso) en lugar de la seguridad gracias al anonimato (que no era el objetivo de RC4). En WEP, RC4 genera un flujo de claves al cual se le aplica un OR exclusivo con el texto llano para dar lugar al texto cifrado.

La carga útil de cada paquete se encripta a través del método de la figura 8-31. Primero se realiza una suma de verificación de la carga útil utilizando el CRC-32 polinomial y la suma de verificación se agrega a la carga útil para formar el texto llano para el algoritmo de encriptación. A continuación, a este texto llano se le aplica un OR exclusivo con un fragmento de flujo de claves de su propio tamaño. El resultado es el texto cifrado. El IV utilizado para iniciar el RC4 se envía junto con el texto cifrado. Cuando el receptor obtiene el paquete, extrae la carga útil de él, genera el flujo de claves a partir de la clave secreta compartida y el IV que acaba de recibir, y aplica un OR exclusivo al flujo de claves con la carga útil para recuperar el texto llano. A continuación puede comprobar la suma de verificación para saber si el paquete fue modificado.

Aunque esta estrategia parece buena a primera vista, se ha publicado un método para violarla (Borisov y cols., 2001). A continuación resumiremos sus resultados. Primero que nada, sorprendentemente muchas instalaciones utilizan la misma clave compartida para todos los usuarios, en cuyo caso cada usuario puede leer todo el tráfico de los demás usuarios. Esto ciertamente es equivalente a Ethernet, pero no es muy seguro.

Pero incluso si cada usuario tiene una clave distinta, WEP aún puede ser atacado. Puesto que por lo general las claves son estables por largos periodos, el estándar WEP recomienda (no obliga a) que el IV se cambie en cada paquete para evitar los ataques de reutilización de flujo de claves

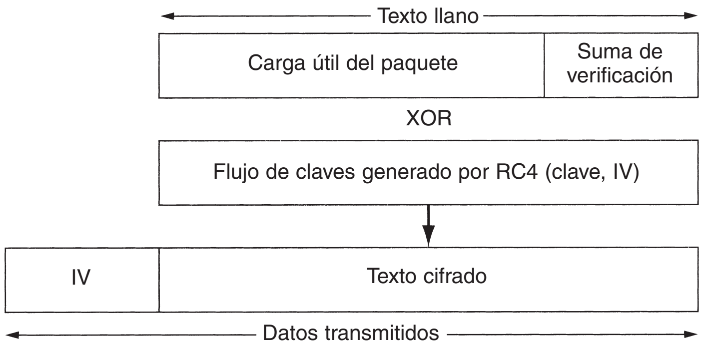


Figura 8-31. Encriptación de paquetes mediante WEP.

que analizamos en la sección 8.2.3. Desgraciadamente, muchas tarjetas 802.11 para computadoras portátiles restablecen el IV a 0 cuando la tarjeta se introduce en la computadora, y lo incrementan en uno por cada paquete enviado. Puesto que las personas remueven e insertan con frecuencia estas tarjetas, son comunes los paquetes con valores bajos de IV. Si Trudy puede coleccionar varios paquetes enviados por el mismo usuario con el mismo valor de IV (que también es enviado en texto llano junto con cada paquete), puede calcular el OR exclusivo de dos valores de texto llano y probablemente romper el cifrado de esa forma.

Pero incluso si la tarjeta 802.11 elige un IV aleatorio para cada paquete, los IVs son de sólo 24 bits, por lo que después de que se hayan enviado 2^{24} paquetes, tienen que reutilizarse los IVs. Peor aún, con los IVs elegidos de manera aleatoria, la cantidad esperada de paquetes que tiene que enviarse antes de que se utilice dos veces el mismo IV es de aproximadamente 5000, debido al ataque de cumpleaños descrito en la sección 8.4.4. Por lo tanto, si Trudy escucha por algunos minutos, es casi seguro que atrape dos paquetes con el mismo IV y la misma clave. Al aplicar un OR exclusivo a los textos cifrados, puede obtener el OR exclusivo de los textos llanos. Esta secuencia de bits puede ser atacada de varias formas para recuperar los textos llanos. Con algo más de trabajo, también puede obtenerse el flujo de claves para ese IV. Trudy puede continuar trabajando de esta manera por un tiempo y compilar un diccionario de flujo de claves para varios IVs. Una vez que se ha descifrado un IV, se pueden desencriptar por completo todos los paquetes que se envíen con él en el futuro (aunque también en el pasado).

Además, puesto que los IVs se utilizan de manera aleatoria, una vez que Trudy ha determinado un par válido (IV, flujo de claves), puede emplearlo para generar todos los paquetes que desee que lo utilicen y, por lo tanto, interferir activamente la comunicación. En teoría, un receptor podría notar que de repente grandes cantidades de paquetes tienen el mismo IV, pero (1) WEP permite esto, y (2) nadie lo verifica.

Por último, el CRC no es de mucha ayuda, puesto que es posible que Trudy cambie la carga útil y haga el cambio correspondiente al CRC, incluso sin tener que eliminar la encriptación. En resumen, es muy sencillo violar la seguridad del 802.11, y no hemos listado todos los ataques que encontraron Borisov y cols.

En agosto de 2001, un mes después de que se presentó el trabajo de Borisov y cols., se publicó otro ataque devastador contra WEP (Fluhrer y cols., 2001). Éste encontró debilidades criptográficas en el RC4 mismo. Fluhrer y cols., descubrieron que muchas de las claves tienen la propiedad de que es posible derivar algunos bits de las claves a partir del flujo de claves. Si este ataque se realiza de manera repetida, es posible deducir toda la clave con un mínimo de esfuerzo. Como el enfoque de su investigación era teórico, Fluhrer y cols., no trataron de romper ninguna LAN 802.11 en la práctica.

En contraste, cuando un estudiante y dos investigadores de los Laboratorios AT&T supieron sobre el ataque de Fluhrer y cols., decidieron llevarlo a la práctica (Stubblefield y cols., 2002). En una semana habían descifrado su primera clave de 128 bits de una LAN 802.11 en funcionamiento, y la mayor parte de esa semana la dedicaron realmente a buscar la tarjeta 802.11 más barata, obtener el permiso para comprarla, instalarla y probarla. La programación tardó sólo dos horas.

Cuando anunciaron sus resultados, la CNN publicó un reportaje en el que algunos gurús de la industria trataron de menospreciar sus resultados afirmando que lo que habían hecho era trivial, pues habían tomado como base el trabajo de Fluhrer y cols. Si bien ese comentario es técnicamente cierto, lo relevante es que los esfuerzos combinados de estos dos equipos demostraron un defecto fatal en WEP y el 802.11.

El 7 de septiembre de 2001, el IEEE respondió al hecho de que WEP se había roto por completo emitiendo una corta declaración en la que señalaba seis puntos que pueden resumirse de la siguiente manera:

1. Les dijimos que la seguridad de WEP no era mejor que la de Ethernet.
2. Es mucho peor olvidarse de establecer alguna clase de seguridad.
3. Traten de utilizar otro tipo de seguridad (por ejemplo, seguridad en la capa de transporte).
4. La siguiente versión, 802.11i, tendrá mejor seguridad.
5. La certificación futura requerirá el uso del 802.11i.
6. Trataremos de determinar qué hacer en tanto llega el 802.11i.

Hemos analizado detenidamente esta historia para resaltar el hecho de que no es sencillo conseguir la seguridad correcta, incluso para los expertos.

Seguridad de Bluetooth

Bluetooth tiene un rango considerablemente más corto que el 802.11, por lo que no puede atacarse desde un estacionamiento, pero la seguridad sigue siendo un problema aquí. Por ejemplo, imagine que la computadora de Alice está equipada con un teclado inalámbrico Bluetooth. En un escenario donde no hubiera seguridad, si Trudy se encontrara en la oficina adyacente, podría leer todo lo que Alice escribiera, incluyendo todo su correo electrónico saliente. También podría capturar todo lo que la computadora de Alice enviara a la impresora Bluetooth instalada junto a ella (por ejemplo, correo electrónico entrante e informes confidenciales). Por fortuna, Bluetooth tiene

un complejo esquema de seguridad para tratar de que todas las Trudies del mundo fracasen. A continuación resumiremos sus características principales.

Bluetooth tiene tres modos de seguridad, que van desde ninguna seguridad en absoluto hasta encriptación completa de datos y control de integridad. Al igual que con el 802.11, si se deshabilita la seguridad (lo predeterminado), no hay seguridad. La mayoría de los usuarios mantiene deshabilitada la seguridad hasta que ocurre una brecha de seguridad; entonces es cuando la habilitan. Este enfoque se parece al dicho "después del niño ahogado, pozo tapado".

Bluetooth proporciona seguridad en múltiples capas. En la capa física, los saltos de frecuencia proporcionan un poco de seguridad, pero debido a que es necesario indicar a cualquier dispositivo Bluetooth de una *piconet* la secuencia de saltos de frecuencia, esta secuencia obviamente no es un secreto. La seguridad real inicia cuando el esclavo recién llegado pide un canal al maestro. Se da por hecho que los dos dispositivos comparten una clave secreta establecida con anticipación. En algunos casos, el fabricante es quien las incluye (por ejemplo, para un teléfono móvil con auriculares vendidos como una sola unidad). En otros casos, un dispositivo (por ejemplo, los auriculares) tiene una clave integrada y el usuario tiene que introducir esa clave en el otro dispositivo (por ejemplo, el teléfono móvil) como un número decimal. Estas claves compartidas se conocen como **claves maestras.**

Para establecer un canal, tanto el esclavo como el maestro verifican si el otro conoce la clave maestra. De ser así, negocian si ese canal será encriptado, si se va a controlar su integridad, o ambas cosas. Después pueden seleccionar una clave de sesión aleatoria de 128 bits, de los cuales algunos pueden ser públicos. El objetivo de permitir esta debilidad de clave es respetar las restricciones gubernamentales de varios países diseñadas para evitar la exportación o el uso de claves más grandes de lo que el gobierno puede romper.

La encriptación utiliza un cifrado de flujo llamado $\mathbf{E_0}$**;** el control de integridad utiliza **SAFER+.** Ambos son cifrados en bloque de clave simétrica tradicionales. SAFER+ fue emitido para el AES *bake-off*, pero se eliminó en la primera ronda porque era más lento que los otros candidatos. Bluetooth se terminó antes de que se eligiera el cifrado AES; de lo contrario éste hubiera utilizado probablemente Rijndael.

En la figura 8-14 se muestra la encriptación real que utiliza el cifrado de flujo, con el texto llano al cual se le aplica OR exclusivo con el flujo de claves para generar el texto cifrado. Desafortunadamente, E_0 mismo (al igual que RC4) podría tener debilidades fatales (Jakobsson y Wetzel, 2001). Si bien no ha sido roto al tiempo de escribir esto, sus similitudes con el cifrado A5/1, cuya falla espectacular puso en peligro el tráfico telefónico de GSM, son causa de preocupación (Biryukov y cols., 2000). Algunas veces sorprende a toda la gente (incluyendo al autor), que en el eterno juego del gato y el ratón entre los criptógrafos y los criptoanalistas, estos últimos salen ganando con mucha frecuencia.

Otro problema de seguridad es que Bluetooth sólo autentica dispositivos, no usuarios, por lo que el robo de un dispositivo Bluetooth podría conceder acceso al ladrón a la cuenta financiera del usuario. Sin embargo, Bluetooth también implementa seguridad en las capas superiores, por lo que incluso en el caso de una brecha de seguridad en el nivel de enlace, podría permanecer algo de seguridad, en especial en las aplicaciones que requieren que se introduzca manualmente un código PIN desde algún tipo de teclado para completar la transacción.

Seguridad de WAP 2.0

En su mayor parte, el foro WAP aprendió su lección al tener una pila de protocolos no estándar en WAP 1.0. En su mayor parte, WAP 2.0 utiliza protocolos estándares en todas las capas. La seguridad no es una excepción. Puesto que está basado en el IP, soporta el uso completo de IPsec en la capa de red. En la capa de transporte, las conexiones TCP pueden protegerse mediante TLS, un estándar IETF que analizaremos más adelante en este capítulo. Más arriba todavía, utiliza autenticación de cliente HTTP, como se define en el RFC 2617. Las crypto bibliotecas a nivel de aplicación proporcionan control de integridad y de no repudio. Después de todo, puesto que WAP 2.0 se basa en estándares bien conocidos, hay una posibilidad de que sus servicios de seguridad —en particular, privacidad, autenticación, control de integridad y no repudio— sean mejores que la seguridad del 802.11 y Bluetooth.

8.7 PROTOCOLOS DE AUTENTICACIÓN

La **autenticación** es la técnica mediante la cual un proceso verifica que su compañero de comunicación sea quien se supone que debe ser y no un impostor. Verificar la identidad de un proceso remoto en la presencia de un intruso activo y malicioso es sorprendentemente difícil y requiere protocolos complejos con base en la criptografía. En esta sección estudiaremos algunos de los muchos protocolos de autenticación que se utilizan en redes de computadoras no seguras.

Además, algunas personas confunden la autorización con la autenticación. Esta última tiene que ver con la interrogante de si usted se está comunicando con un proceso específico. La autorización tiene que ver con lo que ese proceso tiene permitido hacer. Por ejemplo, un proceso cliente contacta un servidor de archivos y dice: Soy el proceso de Scott y deseo eliminar el archivo *cookbook.old*. Desde el punto de vista del servidor, deben contestarse dos preguntas:

1. ¿Éste es el proceso real de Scott (autenticación)?
2. ¿Scott tiene permitido eliminar *cookbook.old* (autorización)?

Sólo después de que estas preguntas se contestan afirmativamente y sin ambigüedad, se puede realizar la acción solicitada. La primera pregunta es la clave. Una vez que el servidor de archivos sabe con quién está hablando, verificar la autorización es sólo cuestión de buscar entradas en las tablas locales o en las bases de datos. Por esta razón, en esta sección nos concentraremos en la autenticación.

Este modelo general es el que utilizan todos los protocolos de autenticación. Alice inicia enviando un mensaje ya sea a Bob o a un **KDC** (**Centro de Distribución de Claves**) confiable, el cual se espera sea honesto. A continuación siguen otros intercambios de mensajes en varias direcciones. Conforme se envían estos mensajes, Trudy podría interceptarlos, modificarlos o repetirlos para engañar a Alice y a Bob o simplemente dañar el trabajo.

Sin embargo, cuando el protocolo se haya completado, Alice está segura de que está hablando con Bob y Bob está seguro de que está hablando con Alice. Asimismo, en la mayoría de los protocolos, dos de ellos también habrán establecido una **clave de sesión** secreta para utilizarla en la próxima conversación. En la práctica, por razones de rendimiento, todo el tráfico de datos se encripta utilizando criptografía de clave simétrica (por lo general, AES o triple DES), aunque la criptografía de clave pública se utiliza ampliamente para los protocolos de autenticación mismos y para establecer la clave de sesión.

El objetivo de utilizar una nueva clave de sesión elegida al azar para cada nueva conexión es minimizar la cantidad de tráfico que se envía con las claves secretas o públicas del usuario, para reducir la cantidad de texto cifrado que un intruso puede obtener, y para minimizar el daño hecho si un proceso falla y su vaciado del núcleo cae en manos equivocadas. Por fortuna, la única clave presente será la de sesión. Todas las claves permanentes deben ponerse en cero con cuidado después de que se haya establecido la sesión.

8.7.1 Autenticación basada en una clave secreta compartida

Para nuestro primer protocolo de autenticación daremos por hecho que Alice y Bob ya comparten una clave secreta, K_{AB}. Esta clave compartida podría haberse acordado por teléfono o en persona, pero no en la red (insegura).

Este protocolo se basa en un principio encontrado en muchos protocolos de autenticación: una parte envía un número aleatorio a la otra, quien a continuación lo transforma en una forma especial y después regresa el resultado. Tales protocolos se conocen como de **desafío-respuesta.** En éste y en los protocolos de autenticación subsiguientes se utilizará la notación que se muestra a continuación:

A, B son las identidades de Alice y Bob.
R_i son los desafíos, donde el subíndice es el retador.
K_i son claves, donde i es el dueño.
K_S es la clave de sesión.

La secuencia de mensajes de nuestro primer protocolo de autenticación de clave compartida se ilustra en la figura 8-32. En el mensaje 1, Alice envía su identidad, A, a Bob en una forma que él entiende. Por supuesto, Bob no tiene forma de saber si este mensaje proviene de Alice o de Trudy, por lo que elige un desafío, un número aleatorio grande, R_B, y lo envía a “Alice” como mensaje 2, en texto llano. Los números aleatorios utilizados una sola vez en los protocolos de desafío-respuesta como éste se conocen como **marcas aleatorias** (***nonces***)**.** A continuación Alice encripta el mensaje con la clave que comparte con Bob y envía el texto cifrado, $K_{AB}(R_B)$, como mensaje 3. Cuando Bob ve este mensaje, inmediatamente sabe que proviene de Alice porque Trudy no conoce K_{AB} y, por lo tanto, no pudo haberlo generado. Además, puesto que R_B fue elegido de manera aleatoria a partir de un espacio grande (digamos, números aleatorios de 128 bits), no es probable que Trudy haya visto R_B y su respuesta en una sesión anterior. Tampoco es probable que pueda adivinar la respuesta correcta de cualquier desafío.

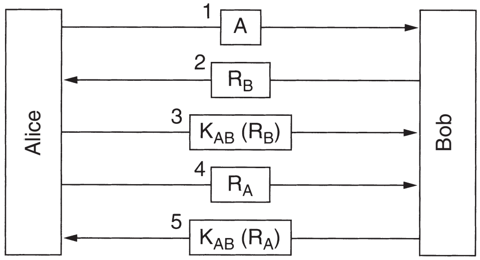


Figura 8-32. Autenticación de dos vías que utiliza un protocolo de desafío-respuesta.

En este punto, Bob está seguro de que está hablando con Alice, pero ella no está segura de nada. Por lo que Alice sabe, Trudy podría haber interceptado el mensaje 1 y regresar R_B como respuesta. Tal vez Bob murió la noche anterior. Para averiguar con quién está hablando, Alice elige un número al azar, R_A y lo envía a Bob como texto llano, en el mensaje 4. Cuando Bob responde con $K_{AB}(R_A)$, Alice sabe que está hablando con Bob. Si desean establecer una clave de sesión ahora, Alice puede elegir una, K_S, encriptarla con K_{AB} y enviarla a Bob.

El protocolo de la figura 8-32 contiene cinco mensajes. Veamos si podemos ser ingeniosos para eliminar algunos de ellos. En la figura 8-33 se muestra un método. Aquí Alice inicia el protocolo de desafío-respuesta en lugar de esperar a que Bob lo haga. De manera similar, mientras Bob responde al desafío de Alice, envía el suyo. El protocolo puede reducirse a tres mensajes en lugar de a cinco.

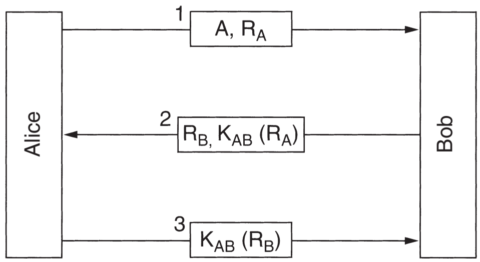


Figura 8-33. Un protocolo de autenticación de dos vías acortado.

¿Este nuevo protocolo es una mejora del original? En un sentido sí lo es: es más corto. Desgraciadamente, también es incorrecto. Bajo algunas circunstancias, Trudy puede vencer este protocolo utilizando lo que se conoce como un **ataque de reflexión.** En particular, Trudy puede romperlo si es posible para abrir a la vez múltiples sesiones con Bob. Por ejemplo, esta situación podría ocurrir si Bob es un banco y está preparado para aceptar muchas conexiones simultáneas de cajeros automáticos.

En la figura 8-34 se muestra el ataque de reflexión de Trudy. Inicia cuando Trudy afirma que es Alice y envía R_T. Bob responde, como es usual, con su propio desafío, R_B. Ahora Trudy está atorada. ¿Qué puede hacer? No conoce K_{AB} (R_B).

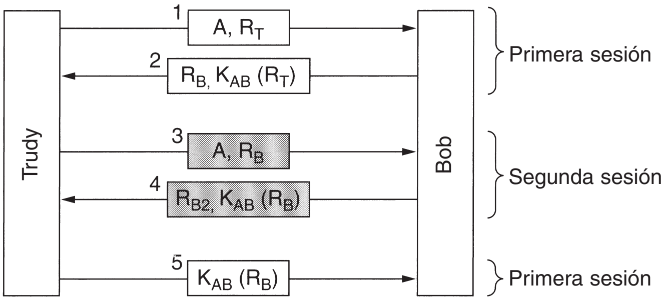


Figura 8-34. El ataque de reflexión.

Trudy puede abrir una segunda sesión con el mensaje 3, y proporcionar un R_B, tomado del mensaje 2, como su desafío. Bob lo encripta con calma y regresa K_{AB} (R_B) en el mensaje 4. Sombreamos los mensajes de la segunda sesión para que resalten. Ahora Trudy tiene la información que le faltaba, por lo que puede completar la primera sesión y abortar la segunda. Bob ahora está convencido de que Trudy es Alice, por lo que cuando ella le pide su estado de cuenta bancaria, Bob se lo proporciona sin más. Después, cuando ella le pide que transfiera todo su dinero a una cuenta secreta en un banco de Suiza, lo hace sin titubeo alguno.

La moraleja de esta historia es:

Diseñar un protocolo de autenticación correcto es más difícil de lo que parece.

Las siguientes cuatro reglas generales con frecuencia ayudan:

1. Obligue al iniciador a que pruebe que es quien dice ser antes de que el contestador tenga que hacerlo. En este caso, Bob reveló información valiosa antes de que Trudy proporcionara cualquier evidencia de que era quien decía ser.

2. Obligue a que tanto el iniciador como el contestador utilicen claves diferentes para probar, aunque esto signifique tener dos claves compartidas, K_{AB} y K'_{AB}.

3. Obligue a que el iniciador y el contestador utilicen conjuntos diferentes para elaborar sus desafíos. Por ejemplo, el iniciador debe utilizar números pares y el contestador números impares.

4. Haga que el protocolo resista a ataques que involucren una segunda sesión paralela en la que la información obtenida en una sesión se utilice en una diferente.

Si se viola alguna de estas reglas, el protocolo puede romperse con frecuencia. Aquí, se violaron las cuatro reglas, con consecuencias desastrosas.

Ahora regresemos y demos un vistazo más de cerca a la figura 8-32. ¿Existe la seguridad de que este protocolo no está sujeto a un ataque de reflexión? Bueno, depende. Es muy sutil. Trudy fue capaz de derrotar nuestro protocolo utilizando un ataque de reflexión porque fue posible abrir una segunda sesión con Bob y engañarlo para que contestara sus propias preguntas. ¿Qué habría pasado si Alice fuera una computadora de propósito general que también aceptara sesiones múltiples, en lugar de una persona en una computadora? Veamos lo que puede hacer Trudy.

Para saber cómo funciona el ataque de Trudy, vea la figura 8-35. Alice inicia anunciando su identidad en el mensaje 1. Trudy intercepta ese mensaje y comienza su propia sesión con el mensaje 2, afirmando que es Bob. Nuevamente sombreamos los mensajes de la sesión 2. Alice responde al mensaje 2 diciendo "¿Dices ser Bob? Pruébalo." en el mensaje 3. En este punto Trudy está atorada porque no puede probar que es Bob.

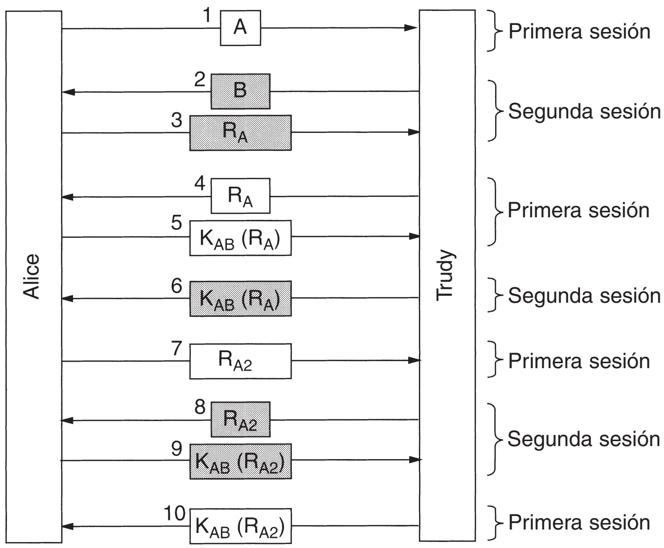


Figura 8-35. Un ataque de reflexión al protocolo de la figura 8-32.

¿Qué hace Trudy ahora? Regresa a la primera sesión, en donde le toca enviar un desafío, y envía el R_A que obtuvo en el mensaje 3. Alice responde amablemente a él en el mensaje 5, y de esta forma proporciona a Trudy la información que necesita para enviar el mensaje 6 en la sesión 2. En este punto, Trudy está prácticamente del otro lado porque ha respondido exitosamente al desafío de Alice en la sesión 2. Ahora puede cancelar la sesión 1, enviar cualquier número antiguo por el resto de la sesión 2 y tendrá una sesión autenticada con Alice en la sesión 2.

Pero Trudy es vil y realmente desea insistir. En lugar de enviar cualquier número antiguo en toda la sesión 2, espera hasta que Alice envía en el mensaje 7, el desafío de Alice para la sesión 1. Por supuesto, Trudy no sabe cómo responder, por lo que utiliza nuevamente el ataque de reflexión, regresando R_{A2} como el mensaje 8. Alice encripta de manera apropiada R_{A2} en el mensaje 9. Trudy

ahora cambia a la sesión 1 y envía a Alice el número que desea en el mensaje 10, copiado de manera conveniente a partir de lo que Alice envió en el mensaje 9. En este punto Trudy tiene dos sesiones completamente autenticadas con Alice.

Este ataque tiene un resultado ligeramente diferente que el ataque del protocolo de tres mensajes que se muestra en la figura 8-34. Esta vez, Trudy tiene dos conexiones autenticadas con Alice. En el ejemplo anterior, tenía una conexión autenticada con Bob. Aquí, si hubiéramos aplicado todas las reglas de protocolos de autenticación analizadas previamente, este ataque podría haberse detenido. Un análisis detallado de este tipo de ataques y cómo frustrarlos se da en (Bird y cols., 1993). También muestran cómo es posible construir de manera sistemática protocolos que tengan altas probabilidades de ser correctos. No obstante, el protocolo más simple de este tipo es un poco complicado, por lo que a continuación mostraremos una clase diferente de protocolo que también funciona.

El nuevo protocolo de autenticación se muestra en la figura 8-36 (Bird y cols., 1993). Utiliza un HMAC del tipo que vimos cuando estudiamos IPsec. Alice inicia enviando a Bob una marca aleatoria, R_A como mensaje 1. Bob responde seleccionando su propia marca aleatoria, R_B, y enviándola junto con un HMAC. El HMAC se forma para construir una estructura de datos que consiste en la marca aleatoria de Alice, la marca aleatoria de Bob, sus identidades y la clave secreta compartida, K_{AB}. A continuación a estos datos estructurados se les aplica un *hash* en el HMAC, por ejemplo utilizando SHA-1. Cuando Alice recibe el mensaje 2, tiene una R_A (que ella misma eligió), R_B, que llega como texto llano, las dos identidades y la clave secreta, K_{AB}, que ha sabido todo el tiempo, por lo que ella misma puede calcular el HMAC. Si corresponde con el HMAC del mensaje, Alice sabe que está hablando con Bob porque Trudy no conoce K_{AB} y, por lo tanto, no sabe cuál HMAC enviar. Alice responde a Bob con un HMAC que contiene sólo las dos marcas aleatorias.

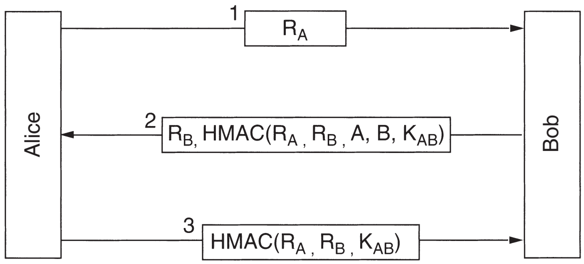


Figura 8-36. Autenticación mediante HMACs.

¿Trudy puede romper de alguna forma este protocolo? No, porque ella no puede obligar a ninguna de las dos partes a encriptar o aplicar un *hash* a un valor de su elección, como sucede en las figuras 8-34 y 8-35. Ambos HMACs incluyen valores elegidos por la parte emisora, algo que Trudy no puede controlar.

El uso de HMACs no es la única forma de utilizar esta idea. Un esquema alternativo que se utiliza con mucha frecuencia en lugar de calcular el HMAC por sobre una serie de elementos es encriptar dichos elementos de manera secuencial utilizando encadenamiento de bloques de cifrado.

8.7.2 Establecimiento de una clave compartida: el intercambio de claves de Diffie-Hellman

Hasta ahora hemos supuesto que Alice y Bob comparten una clave secreta. Suponga que no lo hacen (porque hasta el momento no hay una PKI aceptada universalmente para firmar y distribuir certificados). ¿Cómo pueden establecer una? Una forma sería que Alice llamara a Bob y le diera su clave por teléfono, pero probablemente él iniciaría diciendo: ¿Cómo sé que eres Alice y no Trudy? Podrían tratar de establecer una reunión, en la que cada uno trajera un pasaporte, una licencia de conducir y tres tarjetas de crédito, pero al ser gente ocupada, tal vez no encuentren una fecha aceptable para los dos en meses. Por fortuna, tan increíble como pueda parecer, hay una forma para que personas totalmente extrañas establezcan una clave secreta compartida a la vista de todos, incluso si Trudy está grabando con cuidado cada mensaje.

El protocolo que permite que extraños establezcan una clave secreta compartida se conoce como **intercambio de claves de Diffie-Hellman** (Diffie y Hellman, 1976) y funciona como sigue. Alice y Bob tienen que estar de acuerdo en dos números grandes, n y g, donde n es un número primo, $(n - 1)/2$ también es un número primo y ciertas condiciones se aplican a g. Estos números podrían ser públicos, por lo que cualquiera de ellos simplemente pueden elegir n y g y decirle al otro de manera abierta. Ahora Alice elige un número grande (digamos, de 512 bits), x, y lo mantiene en secreto. De manera similar, Bob elige un número secreto grande, y.

Alice inicia el protocolo de intercambio de claves enviando a Bob un mensaje que contiene (n, g, g^x mod n), como se muestra en la figura 8-37. Bob responde enviando a Alice un mensaje que contiene g^y mod n. Ahora Alice eleva a la x potencia módulo n el número que Bob le envió para obtener $(g^y \text{ mod } n)^x$ *mod n*. Bob realiza una operación similar para obtener $(g^x \text{ mod } n)^y$ *mod n*. Por las leyes de la aritmética modular, ambos cálculos dan como resultado g^{xy} *mod n*. Como por arte de magia, Alice y Bob comparten una clave secreta, g^{xy} mod n.

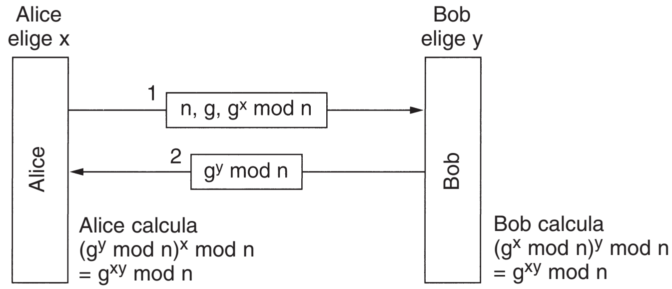


Figura 8-37. El intercambio de claves de Diffie-Hellman.

Por supuesto, Trudy ha visto ambos mensajes. Ella conoce g y n a partir del mensaje 1. Si pudiera calcular x y y, podría adivinar la clave secreta. El problema es que, con sólo g^x mod n, no puede encontrar x. No se conoce ningún algoritmo práctico para calcular logaritmos discretos módulo un número primo muy grande.

Para que el ejemplo sea más concreto, utilizaremos los valores (que no son reales en absoluto) de $n = 47$ y $g = 3$. Alice elige $x = 8$ y Bob elige $y = 10$. Esto se mantiene en secreto. El mensaje de Alice para Bob es (47, 3, 28) porque 3^8 mod 47 es 28. El mensaje de Bob para Alice es (17). Alice calcula 17^8 mod 47, lo cual es 4. Bob calcula 28^{10} mod 47, lo cual es 4. Alice y Bob han determinado de manera independiente que la clave secreta ahora es 4. Trudy tiene que resolver la ecuación 3^x mod 47 = 28, lo cual puede hacerse mediante una búsqueda minuciosa de números pequeños como éstos, pero no cuando todos los números tienen una longitud de cientos de bits. Todos los algoritmos conocidos en la actualidad simplemente tardan mucho, incluso en supercomputadoras paralelas masivas.

A pesar de la elegancia del algoritmo de Diffie-Hellman, hay un problema: cuando Bob obtiene el triple (47, 3, 28), ¿cómo sabe que proviene de Alice y no de Trudy? No hay forma de que pueda saberlo. Desgraciadamente, Trudy puede explotar este hecho para engañar tanto a Alice como a Bob, como se ilustra en la figura 8-38. Aquí, mientras Alice y Bob están eligiendo x y y, respectivamente, Trudy elige su propio número aleatorio, z. Alice envía el mensaje 1 dirigido a Bob. Trudy lo intercepta y envía el mensaje 2 a Bob, utilizando los valores g y n correctos (que de todas formas son públicos) pero con su propio valor z en lugar de x. También regresa el mensaje 3 a Alice. Más tarde Bob envía el mensaje 4 a Alice, el cual es interceptado y conservado nuevamente por Trudy.

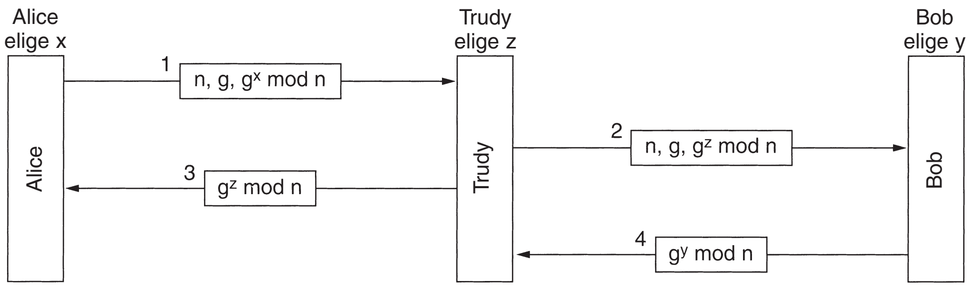


Figura 8-38. El ataque de la brigada de bomberos o de hombre en medio.

Ahora todos realizan la aritmética modular. Alice calcula la clave secreta como g^{xz} mod n, al igual que Trudy (para mensajes destinados a Alice). Bob calcula g^{yz} mod n al igual que Trudy (para mensajes destinados a Bob). Alice piensa que está hablando con Bob por lo que establece una clave de sesión (con Trudy). Bob también lo hace. Cada mensaje que Alice envía en la sesión encriptada, Trudy lo captura, almacena, modifica si lo desea, y lo pasa (lo cual es opcional) a Bob. De manera similar, en la otra dirección. Trudy ve todo y puede modificar todos los mensajes a voluntad, mientras que Alice y Bob tienen la creencia de que tienen un canal seguro. Este ataque se conoce como **ataque de la brigada de bomberos,** porque se parece vagamente a un antiguo departamento de bomberos voluntarios en el que éstos pasan cubetas en fila desde el camión de bomberos hacia el fuego. También se conoce como **ataque de hombre en medio.**

8.7.3 Autenticación que utiliza un centro de distribución de claves

El secreto compartido con un extraño casi funcionó, pero no del todo. Por otro lado, tal vez no valga la pena hacerlo. Para hablar de esta manera con *n* personas, podría necesitar *n* claves. Para las personas populares, la administración de claves podría volverse una verdadera carga, especialmente si cada clave tiene que almacenarse aparte en una tarjeta de chips de plástico.

Un método diferente es introducir un KDC (centro de distribución de claves confiable.) En este modelo, cada usuario tiene una clave compartida con el KDC. Ahora el KDC realiza la administración de claves de sesión y autenticación.

En la figura 8-39 se muestran el protocolo de autenticación KDC más simple conocido que involucra dos partes y un KDC confiable.

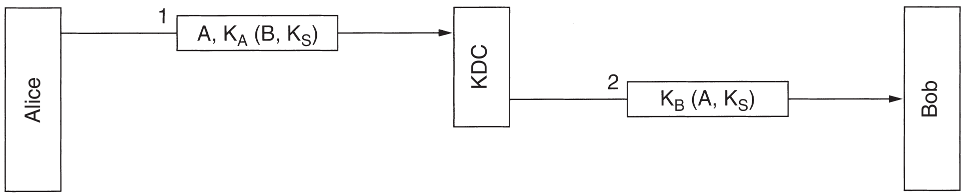


Figura 8-39. Un primer intento en un protocolo de autenticación que utiliza un KDC.

La idea detrás de este protocolo es simple: Alice elige una clave de sesión, K_S, e indica al KDC que desea hablar con Bob utilizando K_S. Este mensaje se encripta con la clave secreta que Alice (sólo) comparte con el KDC, K_A. El KDC desencripta este mensaje, extrayendo la identidad de Bob y la clave de sesión. Después construye un nuevo mensaje que contiene la identidad de Alice y la clave de sesión y envía este mensaje a Bob. Esta encriptación se realiza con K_B, la clave secreta que Bob comparte con el KDC. Cuando Bob desencripta el mensaje, sabe que Alice desea hablar con él y cuál clave desea utilizar.

La autenticación sucede aquí sin costo alguno. El KDC sabe que el mensaje 1 debe provenir de Alice, puesto que nadie más podría encriptarlo con la clave secreta de Alice. De manera similar, Bob sabe que el mensaje 2 debe provenir del KDC, en quien él confía, puesto que nadie más sabe su clave secreta.

Desgraciadamente, este protocolo tiene un defecto importante. Trudy necesita algo de dinero, por lo que idea algún servicio legítimo que puede realizar por Alice, realiza una oferta atractiva y obtiene el trabajo. Después de realizar el trabajo, Trudy educadamente solicita a Alice que pague por transferencia bancaria. A continuación Alice establece una clave de sesión con su banquero, Bob. Después envía a Bob un mensaje en el que le pide que transfiera dinero a la cuenta de Trudy.

Mientras tanto, Trudy regresa a sus antiguos hábitos, espiar la red. Copia los dos mensajes de la figura 8-39 y la solicitud de transferencia de dinero que le sigue. Más tarde, lo repite para Bob. Éste los obtiene y piensa: Alice debe haber contratado nuevamente a Trudy. Ésta seguramente

hace un magnífico trabajo. Después Bob transfiere una cantidad igual de dinero de la cuenta de Alice a la de Trudy. Algún tiempo después del quincuagésimo par de mensajes, Bob sale corriendo de la oficina a fin de encontrar a Trudy para ofrecerle un gran préstamo de manera que pueda ampliar su obviamente exitoso negocio. Este problema se conoce como el **ataque de repetición.**

Son posibles algunas soluciones al ataque de repetición. La primera es incluir una marca de tiempo en cada mensaje. Después, si cada quien recibe un mensaje obsoleto, puede descartarse. El problema con este método es que los relojes en una red nunca están sincronizados, por lo que debe haber un intervalo durante el cual sea válida la marca de tiempo. Trudy puede repetir el mensaje durante este intervalo y marcharse con él.

La segunda solución es colocar una marca aleatoria en cada mensaje. A continuación cada parte tiene que recordar todas las marcas aleatorias previas y rechazar cualquier mensaje que contenga una marca aleatoria utilizada anteriormente. Sin embargo, las marcas aleatorias deben ser recordadas por siempre, a fin de que Trudy no trate de repetir un mensaje de 5 años de antigüedad. Además, si una máquina falla y pierde su lista de marcas aleatorias, nuevamente es vulnerable a un ataque de repetición. Las marcas de tiempo y las marcas aleatorias pueden combinarse para limitar cuánto tiempo deben recordarse las marcas aleatorias, pero claramente el protocolo se va a poner mucho más complicado.

Un método mucho más refinado para la autenticación mutua es utiliza un protocolo de desafío-respuesta de múltiples vías. Un ejemplo bien conocido de tal protocolo es el protocolo de **autenticación de Needham-Schroeder** (Needham y Schroeder, 1978), una variante de lo que se muestra en la figura 8-40.

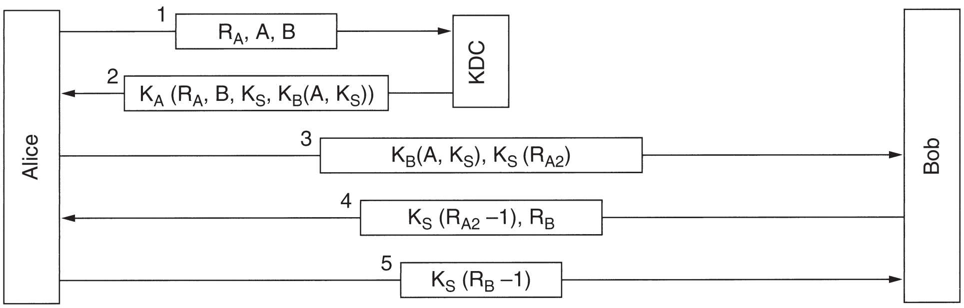


Figura 8-40. El protocolo de autenticación Needham-Schroeder.

El protocolo comienza con Alice diciéndole al KDC que desea hablar con Bob. Este mensaje contiene como marca aleatoria un número aleatorio grande, R_A. El KDC regresa el mensaje 2 que contiene el número aleatorio de Alice, una clave de sesión y un boleto que ella puede regresar a Bob. El objetivo del número aleatorio, R_A, es asegurar a Alice que el mensaje 2 está actualizado y que no es una repetición. La identidad de Bob también está encerrada en caso de que Trudy tenga algunas ideas graciosas sobre reemplazar *B* en el mensaje 1 con su propia identidad a fin de que el

KDC encripte el boleto al final del mensaje 2 con K_T en lugar de K_B. El boleto encriptado con K_B se incluye dentro del mensaje encriptado para evitar que Trudy lo reemplace con algo más cuando el mensaje vaya hacia Alice.

Alice ahora envía el boleto a Bob, junto con un nuevo número aleatorio, R_{A2}, encriptado con la clave de sesión, K_S. En el mensaje 4, Bob regresa K_S $(R_{A2} - 1)$ para probar a Alice que ella está hablando con el Bob verdadero. Regresar K_S (R_{A2}) podría no haber funcionado, puesto que Trudy podría haberlo robado del mensaje 3.

Después de recibir el mensaje 4, Alice está convencida de que está hablando con Bob y de que hasta el momento no es posible que se hayan utilizado repeticiones. Después de todo, simplemente generó R_{A2} algunos milisegundos antes. El propósito del mensaje 5 es convencer a Bob de que sí es Alice con la que está hablando, y de que aquí tampoco se han utilizado repeticiones. Al hacer que las dos partes generen un desafío y una respuesta, se elimina cualquier posibilidad de algún tipo de ataque de repetición.

Aunque este protocolo parece muy sólido, tiene una ligera debilidad. Si Trudy se las arregla para conseguir una sesión de clave antigua en texto llano, puede iniciar una nueva sesión con Bob al repetir el mensaje 3 que corresponde a la clave comprometida y al convencerlo de que ella es Alice (Denning y Sacco, 1981). Esta vez puede saquear la cuenta bancaria de Alice sin tener que realizar ni siquiera una vez el servicio legítimo.

Posteriormente Needham y Schroeder publicaron un protocolo que corrige este problema (Needham y Schroeder, 1987). En el mismo número de la misma publicación, Otway y Rees (1987) también dieron a conocer un protocolo que resuelve el problema en una forma más corta. La figura 8-41 muestra un protocolo Otway-Rees con algunas modificaciones ligeras.

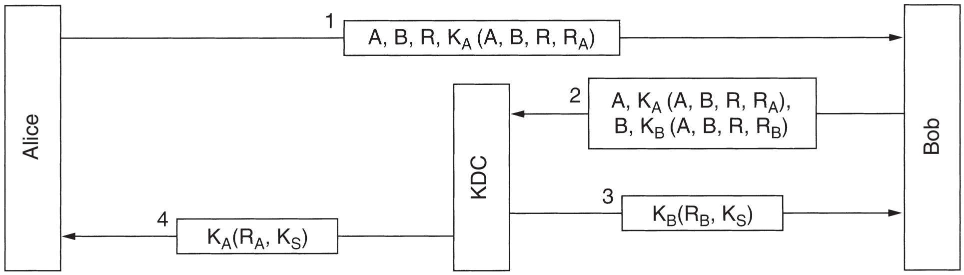


Figura 8-41. El protocolo de autenticación de Otway-Rees (ligeramente simplificado).

En el protocolo de Otway-Rees, Alice inicia generando un par de números aleatorios, R, que se utilizará como identificador común, y R_A, que Alice utilizará para retar a Bob. Cuando Bob obtenga este mensaje, construirá un mensaje nuevo a partir de la parte encriptada del mensaje de Alice y del suyo propio. Ambas partes encriptadas con K_A y K_B identifican a Alice y a Bob, contienen el identificador común y un desafío.

El KDC verifica si en ambas partes R es el mismo. Podría no ser porque Trudy falsificó R en el mensaje 1 o reemplazó parte del mensaje 2. Si los dos Rs coinciden, el KDC cree que el mensaje

de solicitud de Bob es válido. Después genera una clave de sesión y la encripta dos veces, una para Alice y la otra para Bob. Cada mensaje contiene el número aleatorio del receptor, como prueba de que el KDC, y no Trudy, generó el mensaje. En este punto tanto Alice como Bob poseen la misma clave de sesión y pueden comenzar a comunicarse. La primera vez que intercambian mensajes de datos, cada uno puede ver que el otro tiene una copia idéntica de K_S, por lo que la autenticación está completa.

8.7.4 Autenticación utilizando Kerberos

Un protocolo de autenticación utilizado en muchos sistemas reales (incluyendo Windows 2000) es **Kerberos,** que está basado en una variante de Needham-Schroeder. Su nombre proviene del perro de múltiples cabezas de la mitología griega que custodiaba la entrada a Hades (para mantener fuera a las personas indeseables). Kerberos se diseñó en el M.I.T. para permitir que los usuarios de estaciones de trabajo accedieran recursos de red en una forma segura. Su mayor diferencia con respecto de Needham-Schroeder es su suposición de que todos los relojes están bien sincronizados. El protocolo ha pasado por varias iteraciones. V4 es la versión más ampliamente utilizada en la industria, por lo que la describiremos. Más tarde, daremos algunas palabras sobre su sucesor, V5. Para mayor información, vea (Steiner y cols., 1988).

Kerberos involucra a tres servidores además de Alice (una estación de trabajo cliente):

Authentication Server (AS): verifica usuarios durante el inicio de sesión
Ticket-Granting Server (TGS): emite "prueba de boletos de identidad"
Bob, el servidor: hace el trabajo que Alice desea

AS es similar a KDC en que comparte una contraseña secreta con cada usuario. El trabajo de TGS es emitir boletos que puedan convencer a los servidores reales de que el portador de un boleto TGS es realmente quien afirma ser.

Para iniciar una sesión, Alice se sienta frente a una estación de trabajo pública arbitraria y teclea su nombre. La estación de trabajo envía su nombre al AS en texto llano, como se muestra en la figura 8-42. Lo que regresa es una clave de sesión y un boleto, K_{TGS} (A, K_S), destinado para TGS. Estos elementos se empaquetan y encriptan mediante la clave secreta de Alice, de forma que sólo Alice pueda desencriptarlos. Únicamente cuando el mensaje 2 llega, la estación de trabajo pide la contraseña de Alice. A continuación, dicha contraseña se utiliza para generar K_A a fin de desencriptar el mensaje 2 y obtener la clave de sesión y el boleto TGS dentro de él. En este punto, la estación de trabajo sobrescribe la contraseña de Alice para asegurarse de que sólo esté dentro de la estación de trabajo por lo mucho durante algunos milisegundos. Si Trudy intenta iniciar una sesión como Alice, la contraseña que teclee será errónea y la estación de trabajo detectará esto porque la parte estándar del mensaje 2 será incorrecta.

Después de que inicia la sesión, Alice podría decirle a la estación de trabajo que desea contactar a Bob, el servidor de archivos. A continuación la estación de trabajo envía el mensaje 3 al TGS preguntándole por un boleto para utilizar con Bob. El elemento clave en esta petición es K_{TGS} (A, K_S), que se encripta con la clave secreta del TGS y que se utiliza como prueba de que el

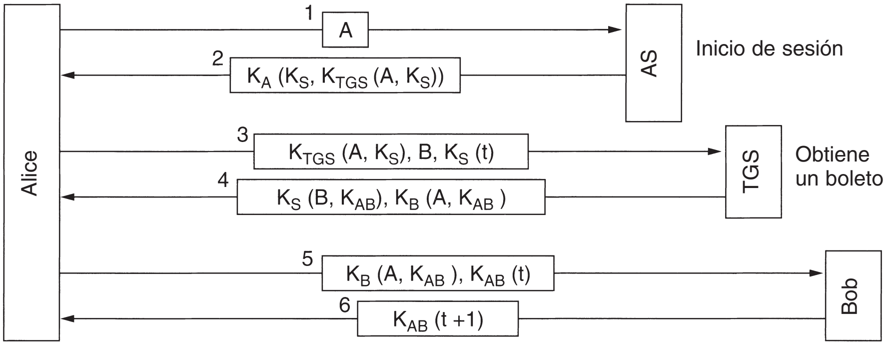


Figura 8-42. El funcionamiento de Kerberos V4.

emisor es realmente Alice. El TGS responde con la creación de una clave de sesión, K_{AB}, para que Alice la utilice con Bob. Se regresan dos versiones de dicha clave. La primera se encripta sólo con K_S, de manera que Alice pueda leerla. La segunda se encripta con la clave de Bob, K_B, de manera que Bob pueda leerla.

Trudy puede copiar el mensaje 3 y tratar de utilizarlo nuevamente, pero no podrá hacerlo porque se lo impedirá la marca de tiempo encriptada, *t*, la cual se envía junto con él. Trudy no puede reemplazar la marca de tiempo con una más reciente, porque ella no conoce K_S, la clave de sesión que Alice utiliza para hablar con el TGS. Incluso si Trudy repite con rapidez el mensaje 3, todo lo que obtendrá será otra copia del mensaje 4, el cual no pudo desencriptar la primera vez y que tampoco podrá desencriptar esta segunda vez.

Ahora Alice puede enviar K_{AB} a Bob para establecer una sesión con él. Este intercambio también tiene establecidas marcas de tiempo. La respuesta es probar a Alice que realmente está hablando con Bob, no con Trudy.

Después de esta serie de intercambios, Alice puede comunicarse con Bob bajo la cobertura de K_{AB}. Si más adelante Alice necesita hablar con otro servidor, Carol, simplemente repite el mensaje 3 al TGS, sólo que ahora especificando *C* en lugar de *B*. El TGS responderá rápidamente con un boleto encriptado con K_C que Alice puede enviar a Carol, y que Carol aceptará como prueba de que proviene de Alice.

El objetivo de todo este trabajo es que ahora Alice puede acceder servidores a través de toda la red de forma segura y que su contraseña no tiene que pasar a través de la red. De hecho, sólo tuvo que estar en su propia estación de trabajo durante algunos milisegundos. Sin embargo, observe que cada servidor realiza su propia autorización. Cuando Alice presenta su boleto a Bob, esto simplemente prueba a Bob quién lo envió. Bob es quien determina las acciones que Alice tiene permitido realizar.

Puesto que los diseñadores de Kerberos no esperaron que todo el mundo confiara en un solo servidor de autenticación, establecieron reglas para tener múltiples **dominios,** cada uno con su propio AS y TGS. Para obtener un boleto para un servidor de un dominio distante, Alice podría solicitar a su propio TGS un boleto que sea aceptado por el TGS en el dominio distante. Si el TGS distante se ha registrado con el TGS local (de la misma manera en que lo hacen los servidores

locales), este último le dará a Alice un boleto válido en el TGS distante. A continuación ella puede hacer negocios, como obtener boletos para servidores de ese dominio. Sin embargo, observe que para que las partes de dos dominios hagan negocios, cada una debe confiar en el TGS de la otra.

Kerberos V5 es más elegante que V4 y tiene más sobrecarga. También utiliza OSI ASN.1 (Notación de Sintaxis Abstracta 1) para describir tipos de datos y tiene pequeños cambios en los protocolos. Además, sus boletos tienen duraciones más largas, permite que los boletos sean renovados y emitirá boletos postfechados. Además, al menos en teoría, no depende de DES, como lo hace V4, y soporta múltiples dominios pues delega la generación de boletos a múltiples servidores de boletos.

8.7.5 Autenticación utilizando criptografía de clave pública

La autenticación mutua también puede realizarse utilizando criptografía de clave pública. Para empezar, Alice necesita obtener la clave pública de Bob. Si existe una PKI con un servidor de directorios que proporciona certificados de clave pública, Alice puede pedir la de Bob, que en la figura 8-43 se muestra como mensaje 1. La repetición, en el mensaje 2, es un certificado X.509 que contiene la clave pública de Bob. Cuando Alice verifica que la firma sea correcta, envía a Bob un mensaje que contiene su identidad y una marca aleatoria.

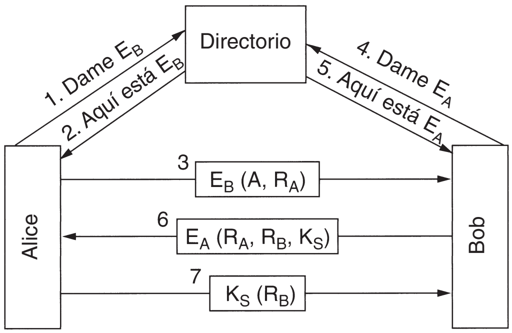


Figura 8-43. Autenticación mutua utilizando la criptografía de clave pública.

Cuando Bob recibe este mensaje, no tiene idea de si proviene de Alice o de Trudy, pero sigue adelante y pide al servidor de directorio la clave pública de Alice (mensaje 4) la cual obtiene de inmediato (mensaje 5). A continuación envía a Alice como mensaje 6 uno que contiene el R_A de Alice, su propia marca aleatoria, R_B, y una clave de sesión propuesta, K_S.

Cuando Alice obtiene el mensaje 6, lo desencripta mediante su clave privada. Ve el R_A en dicho mensaje, lo cual le trae alivio. El mensaje debe provenir de Bob, puesto que Trudy no tiene forma de determinar R_A. Además, debe ser una actualización y no una repetición, puesto que acaba de enviarle a Bob el R_A. Alice accede a la sesión enviando el mensaje 7. Cuando Bob ve R_B encriptado con la clave de sesión que acaba de generar, sabe que Alice obtuvo el mensaje 6 y verificó R_A.

¿Qué puede hacer Trudy para tratar de arruinar este protocolo? Puede fabricar el mensaje 3 y engañar a Bob para que investigue a Alice, pero ésta verá un R_A que ella no envió y ya no proseguirá. Trudy no podrá falsificar el mensaje 7 para Bob porque no conoce R_B o K_S y no puede determinarlos sin la clave privada de Alice. No es el día de suerte de Trudy.

8.8 SEGURIDAD DE CORREO ELECTRÓNICO

Cuando se envía un mensaje de correo electrónico entre dos sitios distantes, por lo general pasará por docenas de máquinas. Cualquiera de ellas puede leer y registrar el mensaje para uso futuro. En la práctica, no existe la privacidad, a pesar de lo que mucha gente piensa. Sin embargo, muchas personas desearían tener la capacidad de enviar correo electrónico que sólo pueda ser leído por el destinatario y por nadie más: ni siquiera su jefe ni su gobierno. Este deseo ha estimulado a varias personas y grupos a que apliquen al correo electrónico los principios criptográficos que estudiamos anteriormente para producir correo electrónico seguro. En las siguientes secciones analizaremos un sistema seguro ampliamente utilizado, PGP, y después mencionaremos brevemente dos más, PEM y S/MIME. Para información adicional sobre el correo electrónico seguro, vea (Kaufman y cols., 2002, y Schneier, 1995).

8.8.1 PGP—Privacidad Bastante Buena

Nuestro primer ejemplo, **PGP** (**Privacidad Bastante Buena**), es esencialmente la idea original de una persona, Phil Zimmermann (Zimmermann, 1995a, 1995b). Zimmermann es un defensor de la privacidad cuyo lema es: Si la privacidad se elimina, solamente los delincuentes tendrán privacidad. Liberado en 1991, PGP es un paquete de seguridad de correo electrónico completo que proporciona privacidad, autenticación, firmas digitales y compresión, todo en una forma fácil de utilizar. Además, todo el paquete, incluyendo todo el código fuente, se distribuye sin costo alguno a través de Internet. Debido a su cualidad, precio (ninguno) y fácil disponibilidad en las plataformas UNIX, Linux, Windows y Mac OS, se utiliza ampliamente hoy en día.

PGP encripta los datos utilizando un cifrado de bloque llamado **IDEA** (**Algoritmo Internacional de Encriptación de Datos**), que utiliza claves de 128 bits. Se diseñó en Suecia en la época en que el DES fue visto como defectuoso y el AES todavía no se inventaba. De manera conceptual, IDEA es similar a DES y AES: mezcla los bits en una serie de rondas, pero los detalles de las funciones de mezcla son diferentes de los de DES y AES. La administración de claves utiliza RSA y la integridad de datos utiliza MD5, temas que ya hemos analizado.

PGP también se ha visto envuelto en controversias desde su primer día (Levy, 1993). Debido a que Zimmermann no hizo nada para que otras personas no colocaran a PGP en Internet, en donde personas de todo el mundo podían accederlo, el gobierno de los Estados Unidos declaró que Zimmermann había violado las leyes de los Estados Unidos que prohibían la exportación de equipo de guerra. La investigación del gobierno de los Estados Unidos en contra de Zimmermann duró cinco años, pero con el tiempo se abandonó, probablemente por dos razones. Primera, Zimmermann

mismo no colocó a PGP en Internet, por lo que su abogado alegó que *Zimmermann* nunca exportó nada (y ahí entra la controversia de si la creación de un sitio Web significa exportación). Segunda, con el tiempo el gobierno se dio cuenta de que ganar el juicio significaba convencer al jurado de que el hecho de crear un sitio Web que contenga un programa de privacidad descargable estaba cubierto por la ley de tráfico de armas que prohíbe la exportación de material de guerra, como tanques, submarinos, aviones militares y armas nucleares. Los años de mala publicidad tampoco fueron de mucha ayuda.

Además, las leyes de exportación son extrañas, por decir lo menos. El gobierno consideró la colocación de código en un sitio Web como una exportación ilegal y persiguió a Zimmermann durante cinco años por eso. Por otro lado, cuando alguien publicó todo el código fuente PGP, en C, en un libro (en una fuente grande con una suma de verificación en cada página para hacer que su digitalización fuera fácil) y después exportó dicho libro, no hubo problema con el gobierno debido a que los libros no se clasifican como equipo de guerra. La espada es más poderosa que la pluma, por lo menos para el Tío Sam.

Otro problema en el que se vio involucrado PGP tuvo que ver con la infracción de las patentes. La compañía que poseía la patente RSA, RSA Security, Inc., alegó que el uso que PGP hacía del algoritmo RSA infringió su patente, pero ese problema quedó resuelto a partir de la versión 2.6. Además, PGP utiliza otro algoritmo de encriptación patentado, IDEA, cuyo uso causó algunos problemas al principio.

Puesto que el código de PGP es abierto, diversas personas y grupos lo han modificado y producido varias versiones. Algunas de ellas se diseñaron para resolver los problemas con las leyes de equipo de guerra, otras se enfocaron en evitar el uso de algoritmos patentados, y otros más desearon convertirlo en un producto comercial de código cerrado. Aunque en la actualidad las leyes de equipo de guerra se han liberado ligeramente (de lo contrario los productos que utilizan AES no podrían exportarse de los Estados Unidos), y la patente de RSA caducó en septiembre de 2000, el legado de todos estos problemas es que hay en circulación varias versiones incompatibles de PGP, bajo diversos nombres. La discusión anterior se enfoca en el PGP clásico, que es la versión más antigua y simple. En el RFC 2440 se describe otra versión popular, Open PGP. Otra más es GNU Privacy Guard.

PGP utiliza de manera intencional algoritmos criptográficos existentes en lugar de inventar nuevos. Se basa principalmente en algoritmos que han aguantado revisiones extensas de iguales y no fueron diseñados o influenciados por ninguna agencia gubernamental para debilitarlos. Para las personas que intentan desconfiar del gobierno, esta propiedad es una gran ganancia.

PGP soporta la compresión de texto, confidencialidad y firmas digitales, además de que proporciona características de administración extensa de claves, pero por extraño que parezca, no proporciona características de correo electrónico. Es más que un preprocesador que toma como entrada texto llano y produce como salida texto cifrado firmado en base64. Por supuesto, después esta salida puede enviarse por correo electrónico. Algunas implementaciones de PGP llaman a un agente de usuario como paso final para enviar realmente el mensaje.

Para ver cómo funciona PGP, consideremos el ejemplo que se muestra en la figura 8-44. Aquí, Alice desea enviar de forma segura un mensaje en texto llano firmado, *P*, a Bob. Tanto Alice

como Bob tienen claves públicas (E_X) y privadas (D_X) RSA. Supongamos que cada uno conoce la clave pública del otro; analizaremos la administración de claves PGP dentro de poco.

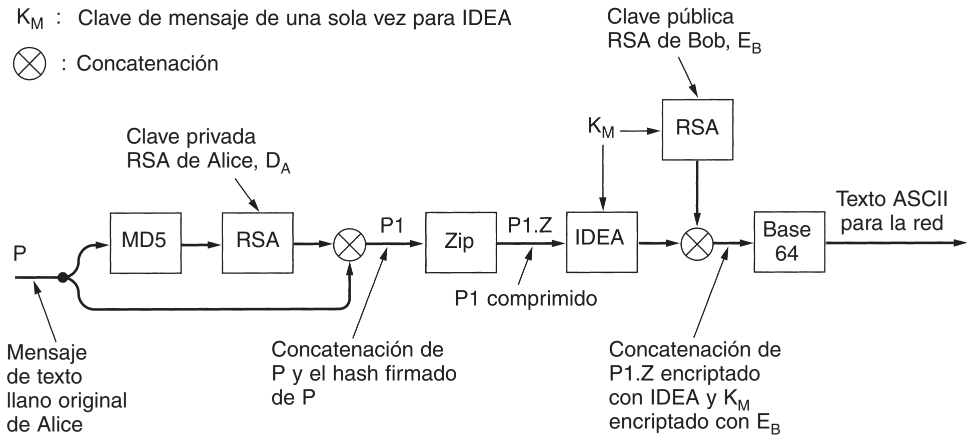


Figura 8-44. PGP en funcionamiento para enviar un mensaje.

Alice empieza por abrir el programa PGP en su computadora. Lo primero que hace PGP es aplicar un *hash* al mensaje, *P*, mediante MD5, y después encripta el *hash* resultante mediante su clave privada RSA, D_A. Cuando Bob obtiene el mensaje, puede desencriptar el *hash* con la clave pública de Alice y verificar que dicho *hash* sea correcto. Incluso si alguien más (por ejemplo, Trudy) pudiera adquirir el *hash* en esta etapa y desencriptarlo con la clave pública conocida de Alice, la fuerza de MD5 garantiza que desde un punto de vista computacional sea imposible producir otro mensaje con el mismo *hash* MD5.

El *hash* encriptado y el mensaje original ahora están concatenados en un solo mensaje, *P1*, y se comprimen mediante un programa ZIP, que utiliza el algoritmo Ziv-Lempel (Ziv y Lempel, 1977). Llame *P1.Z* a la salida de este paso.

A continuación, PGP pide a Alice alguna entrada aleatoria. Tanto el contenido como la velocidad de tecleo se utilizan para generar una clave de mensaje IDEA de 128 bits, K_M (llamada clave de sesión en la literatura de PGP, pero éste realmente es un nombre inapropiado puesto que no es una sesión). K_M ahora se utiliza para encriptar *P*1.*Z* con IDEA en modo de retroalimentación de cifrado. Además, K_M se encripta con la clave pública de Bob, E_B. Estos dos componentes después se concatenan y convierten a base64, como analizamos en la sección sobre MIME en el capítulo 7. El mensaje resultante contiene sólo letras, dígitos y los símbolos +, / e =, lo que significa que puede colocarse en el cuerpo del RFC 822 y se puede esperar que llegue sin modificaciones.

Cuando Bob obtiene el mensaje, invierte la codificación base64 y desencripta la clave IDEA mediante su clave privada RSA. Bob puede utilizar esta clave para desencriptar el mensaje a fin

de obtener *P1.Z*. Después de descomprimirlo, Bob separa el texto llano del *hash* encripta y desencripta este último meditante la clave pública de Alice. Si el *hash* de texto llano coincide con su propio cálculo MD5, Bob sabe que *P* es el mensaje correcto y que proviene de Alice.

Vale la pena señalar que aquí RSA sólo se utiliza en dos lugares: para encriptar el *hash* MD5 de 128 bits y para desencriptar la clave IDEA de 128 bits. Aunque RSA es lento, sólo tiene que encriptar 256 bits, no un gran volumen de datos. Además, los 256 bits de texto llano son en extremo aleatorios, por lo que se necesitará que Trudy realice una cantidad considerable de trabajo para determinar si una clave adivinada es correcta. IDEA realiza la encriptación pesada, que es varios órdenes de magnitud más rápida que RSA. Por lo tanto, PGP proporciona seguridad, compresión y una firma digital y lo hace en una forma más eficiente que el esquema ilustrado en la figura 8-19.

PGP soporta cuatro longitudes de clave RSA. Es responsabilidad del usuario seleccionar la más adecuada. Las longitudes son:

1. Casual (384 bits): En la actualidad puede romperse con facilidad.
2. Commercial (512 bits): Organizaciones de tres letras pueden romperla.
3. Military (1024 bits): Nadie de este mundo puede romperla.
4. Alien (2048 bits): Ni siquiera alguien de otro planeta puede romperla.

Puesto que RSA sólo se utiliza para dos pequeños cálculos, todo mundo debería utilizar las claves de fuerza alien todo el tiempo.

En la figura 8-45 se muestra el formato de un mensaje PGP clásico. También se utilizan otros formatos. El mensaje tiene tres partes, las cuales contienen la clave IDEA, la firma y el mensaje, respectivamente. La parte de la clave no sólo contiene ésta, sino también un identificador de clave, puesto que se permite que los usuarios tengan múltiples claves públicas.

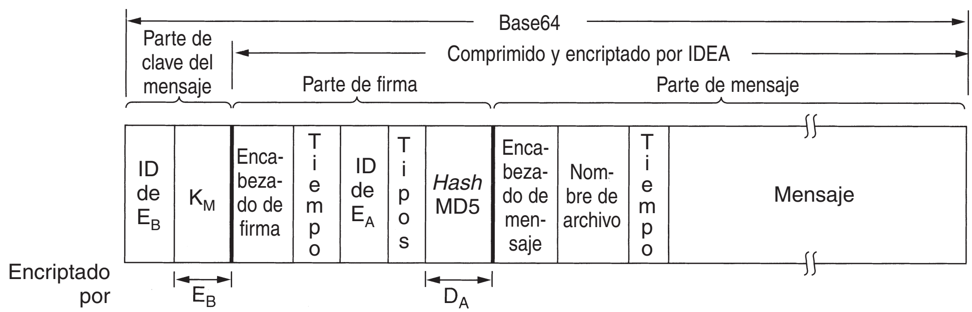


Figura 8-45. Un mensaje PGP.

La parte de firma contiene un encabezado, el cual no trataremos aquí. El encabezado está seguido por una marca de tiempo, el identificador para la clave pública del emisor que puede utili-

zarse para desencriptar el *hash* de firma, alguna información de tipos que identifica los algoritmos utilizados (para permitir que MD6 y RSA2 se puedan utilizar cuando se inventen) y el *hash* encriptado mismo.

La parte del mensaje también contiene un encabezado, el nombre predeterminado del archivo que se va a utilizar si el receptor escribe el archivo en disco, una marca de creación de mensajes y, finalmente, el mensaje mismo.

La administración de claves ha recibido una gran cantidad de atención en PGP puesto que es el talón de Aquiles de los sistemas de seguridad. La administración de claves funciona como sigue. Cada usuario mantiene localmente dos estructuras de datos: un anillo de clave privada y un anillo de clave pública. El **anillo de clave privada** contiene uno o más pares de claves pública-privada personales. La razón para soportar múltiples pares por usuario es permitir que los usuarios cambien sus claves públicas periódicamente o cuando piensen que han sido comprometidas, sin invalidar los mensajes que están actualmente en preparación o en tránsito. Cada par tiene asociado un identificador asociado, por lo que un emisor de mensajes puede indicarle al destinatario cuál clave pública se utilizó para encriptarlo. Los identificadores de mensajes consisten en los 64 bits de orden menor de la clave pública. Los usuarios son responsables de evitar conflictos en sus identificadores de clave pública. Las claves privadas en disco se encriptan mediante una contraseña especial (de tamaño arbitrario) para protegerlas contra ataques de robo.

El **anillo de clave pública** contiene claves públicas de los contactos del usuario. Éstas son necesarias para encriptar las claves de mensaje asociadas con cada mensaje. Cada entrada del anillo de clave pública no sólo contiene la clave pública, sino también su identificador de 64 bits y una indicación de qué tanto confía el usuario en la clave.

El problema que se trata de solucionar aquí es el siguiente. Suponga que las claves públicas se mantienen en boletines electrónicos. Un forma para que Trudy pueda leer el correo electrónico secreto de Bob es atacar el boletín electrónico y reemplazar la clave pública de Bob con una de su elección. Cuando más tarde Alice obtiene la clave que supuestamente pertenece a Bob, Trudy puede atacar a Bob mediante un ataque de brigada de bomberos.

Para evitar tales ataques, o por lo menos minimizar las consecuencias de ellos, Alice necesita saber cuánto debe confiar en el elemento llamado "clave de Bob" de su anillo de clave pública. Si Alice sabe que Bob le proporcionó personalmente un disco flexible que contiene la clave, puede establecer el valor confiable al valor más alto. Este enfoque controlado por el usuario y descentralizado hacia la administración de claves públicas es el que mantiene apartado a PGP de los esquemas centralizados PKI.

Sin embargo, las personas algunas veces obtienen claves públicas cuando consultan al servidor de claves confiable. Por esta razón, después de que X.509 se estandarizó, PGP soportó estos certificados, así como el mecanismo de anillo de clave pública de PGP. Todas las versiones actuales de PGP soportan X.509.

8.8.2 PEM—Correo con Privacidad Mejorada

En contraste con PGP, que inicialmente fue idea de un solo hombre, nuestro segundo ejemplo, **PEM (Correo con Privacidad Mejorada),** desarrollado a finales de la década de 1980, es un

estándar oficial de Internet que se describe en cuatro RFCs: del RFC 1421 al 1424. PEM cubre más o menos el mismo territorio que PGP: privacidad y autenticación para los sistemas de correo electrónico basados en el RFC 822. Sin embargo, también tiene algunas diferencias con respecto a PGP en lo que se refiere a metodología y tecnología.

Los mensajes enviados mediante PEM primero se convierten en una forma canónica a fin de que tengan las mismas convenciones sobre los espacios en blanco (por ejemplo, tabuladores, espacios en blanco). Después se calcula un *hash* de mensaje mediante MD2 o MD5, y la concatenación del *hash* y del mensaje se encripta mediante DES. Puesto que se sabe que una clave de 56 bits es débil, esta opción no es muy conveniente. El mensaje encriptado puede codificarse más tarde con codificación base64 y transmitirse al destinatario.

Al igual que en PGP, cada mensaje se encripta con una clave de una sola vez que se encierra junto con el mensaje. La clave puede protegerse con el RSA o con el DES triple mediante EDE.

La administración de claves está más estructurada que en PGP. Las claves están certificadas por certificados X.509 emitidos por CAs, que están ordenados en una jerarquía rígida que comienza en una sola raíz. La ventaja de este esquema es que es posible la revocación de certificados al hacer que la raíz emita CRLs de manera periódica.

El único problema con PEM es que nadie lo ha utilizado y hace mucho tiempo que se fue al cielo de los bits. El problema es principalmente político: ¿quién podría operar la raíz y bajo qué condiciones? No faltaban candidatos, pero las personas estaban temerosas de confiar a alguna compañía la seguridad de todo el sistema. El candidato más serio, RSA Security, Inc., quería cobrar por certificado emitido. Sin embargo, algunas organizaciones se opusieron a esta idea. En particular, el gobierno de los Estados Unidos tiene permitido utilizar sin costo alguno todas las patentes, y las compañías fuera de Estados Unidos se habían acostumbrado a utilizar de manera gratuita el algoritmo RSA (la compañía olvidó patentarla fuera de Estados Unidos). Nadie estaba feliz por tener que pagar de repente a RSA Security, Inc., por algo que siempre se había realizado de manera gratuita. Al final, no se encontró ninguna raíz y PEM se colapsó.

8.8.3 S/MIME

La siguiente aventura del IETF en lo que se refiere a la seguridad de correo electrónico, llamada **S/MIME** (**MIME Seguro**), se describe en los RFCs 2632 al 2643. Al igual que PEM, proporciona autenticación, integridad de datos, confidencialidad y no repudio. También es muy flexible, pues soporta una variedad de algoritmos criptográficos. No es sorprendente, dado el nombre, que S/MIME se integre bien con MIME, lo cual permite la protección de todos los mensajes. Se define una gran variedad de encabezados MIME, por ejemplo, para contener firmas digitales.

IETF definitivamente aprendió algo de la experiencia de PEM. S/MIME no tiene una jerarquía de certificados rígida que comienza en una sola raíz. En su lugar, los usuarios pueden tener múltiples anclas confiables. Un certificado se considera válido siempre y cuando pueda ser rastreado hacia alguna ancla en la que el usuario confíe. S/MIME utiliza los algoritmos y protocolos estándar que hemos examinado hasta ahora, por lo que no los trataremos aquí. Para más detalles, por favor consulte los RFCs.

8.9 SEGURIDAD EN WEB

Hemos estudiado dos áreas importantes en donde es necesaria la seguridad: las comunicaciones y el correo electrónico. Puede considerarlos como la sopa y el entremés. Ahora es tiempo del platillo principal: la seguridad en Web. Web es el lugar donde la mayoría de las Trudies vagan en la actualidad y tratan de realizar trabajo sucio. En las siguientes secciones analizaremos algunos de los problemas y cuestiones que se relacionan con la seguridad en Web.

La seguridad en Web se puede dividir en tres partes. La primera, ¿cómo se nombran de manera segura los objetos y los recursos? Segunda, ¿cómo pueden establecerse las conexiones seguras y autenticadas? Tercera, ¿qué sucede cuando un sitio Web envía a un cliente una pieza del código ejecutable? Después de ver algunas amenazas, examinaremos todas estas cuestiones.

8.9.1 Amenazas

Casi semanalmente, uno lee en los periódicos los problemas de seguridad en los sitios Web. La situación es muy sombría. Veamos algunos ejemplos de lo que ha sucedido en realidad. Primero, la página de inicio de numerosas organizaciones ha sido atacada y reemplazada por una nueva página de inicio de la elección de los *crackers*. (La prensa popular llama “hackers” a las personas que irrumpen en las computadoras, pero muchos programadores reservan ese término para los grandes programadores. Nosotros preferimos llamar “crackers” a las personas que irrumpen sin permiso a la computadora de otra persona.) Entre los sitios que han sido atacados por los *crackers* se incluyen Yahoo, el Ejército de Estados Unidos, la CIA, la NASA y el *New York Times*. En muchos casos, los *crackers* simplemente colocaron algún texto gracioso y los sitios fueron reparados en algunas horas.

Ahora veamos algunos casos más serios. Muchos sitios han sido derribados por ataques de negación de servicio, en los que el *cracker* inunda con tráfico el sitio, dejándolo incapaz de responder a consultas legítimas. Con frecuencia, el ataque se realiza desde varias máquinas en las que el *cracker* ya ha irrumpido (ataques DDoS). Estos ataques son tan comunes que ya no son noticia, pero pueden costar al sitio atacado miles de dólares en negocios perdidos.

En 1999, un *cracker* sueco irrumpió en el sitio Web Hotmail de Microsoft y creó un sitio espejo que permitió que cualquiera tecleara el nombre de un usuario de Hotmail y leyera todo el correo electrónico de la persona correspondiente.

En otro caso, un *cracker* ruso de 19 años llamado Maxim irrumpió en un sitio Web de comercio y robó 300,000 números de tarjetas de crédito. Después contactó a los dueños del sitio y les dijo que si no le pagaban $100,000, publicaría en Internet todos los números de las tarjetas. Los dueños no cedieron a su chantaje y Maxim publicó dichos números, dañando a muchas víctimas inocentes.

Otro caso fue el de un estudiante de California de 23 años que envió por correo electrónico un comunicado de prensa a una agencia de noticias en el que declaraba falsamente que Emulex Corporation iba a publicar una gran pérdida trimestral y que el director general iba a renunciar

inmediatamente. En pocas horas, las acciones de la compañía cayeron en 60%, causando que los inversionistas perdieran más de $2 mil millones. El autor de esta noticia ganó un cuarto de millón de dólares al vender las acciones poco antes de enviar el anuncio. Si bien este evento no tuvo que ver con que alguien irrumpiera en un sitio Web, es claro que colocar un anuncio de tal magnitud en una página de inicio de cualquier corporación importante podría tener un efecto similar.

(Desgraciadamente) podríamos seguir contando muchos casos como éstos, pero es tiempo de que examinemos algunos de los aspectos técnicos relacionados con la seguridad en Web. Para mayor información sobre los problemas de seguridad de todo tipo, vea (Anderson, 2001; Garfinkel con Spafford, 2002, y Schneier, 2000). En Internet podría encontrar una gran cantidad de casos específicos.

8.9.2 Asignación segura de nombres

Comencemos con algo muy básico: Alice desea visitar el sitio Web de Bob. Ella teclea el URL de Bob en su navegador y segundos más tarde, aparece una página Web. Pero, ¿es la de Bob? Tal vez. Trudy podría estar haciendo sus viejos trucos nuevamente. Por ejemplo, tal vez esté interceptando y examinando todos los paquetes salientes de Alice. Cuando captura una solicitud *GET* de HTTP dirigida al sitio Web de Bob, puede ir ella misma a dicho sitio para obtener la página, modificarla como lo desea y regresar la página falsa a Alice, la cual no sería extraña para Alice. Peor aún, Trudy podría recortar los precios de la tienda electrónica de Bob para hacer que los precios de los productos parezcan muy atractivos, con lo que engañaría a Alice para que enviara su número de tarjeta de crédito a "Bob" para comprar algo de mercancía.

Una desventaja de este clásico ataque de hombre en medio es que Trudy tiene que estar en una posición para interceptar el tráfico saliente de Alice y falsificar su tráfico entrante. En la práctica, tiene que intervenir la línea telefónica de Alice o la de Bob, puesto que intervenir la red dorsal de fibra es mucho más difícil. Aunque intervenir la línea es ciertamente posible, también significa mucho trabajo, y aunque Trudy es inteligente, también es floja. Además, hay formas más fáciles de engañar a Alice.

Falsificación de DNS

Por ejemplo, suponga que Trudy es capaz de violar el sistema DNS, tal vez sólo el caché DNS del ISP de Alice, y reemplazar la dirección IP de Bob (digamos, 36.1.2.3) con la suya propia (digamos, 42.9.9.9). Eso lleva al siguiente ataque. En la figura 8-46(a) se ilustra la forma en que se supone debe trabajar. Aquí (1) Alice pide al DNS la dirección IP de Bob, (2) la obtiene, (3) le pide a Bob su página de inicio, y (4) también la obtiene. Después de que Trudy ha modificado el registro DNS de Bob para contener su propia dirección IP en lugar de la de Bob, obtenemos la situación de la figura 8-46(b). Aquí, cuando Alice busca la dirección IP de Bob, obtiene la de Trudy, por lo que todo su tráfico dirigido a Bob va a parar a las manos de Trudy. Ahora, ella puede iniciar un ataque de hombre en medio sin tener que intervenir ninguna línea telefónica. En su lugar, tiene que irrumpir en un servidor DNS y cambiar un registro, lo cual es más fácil.

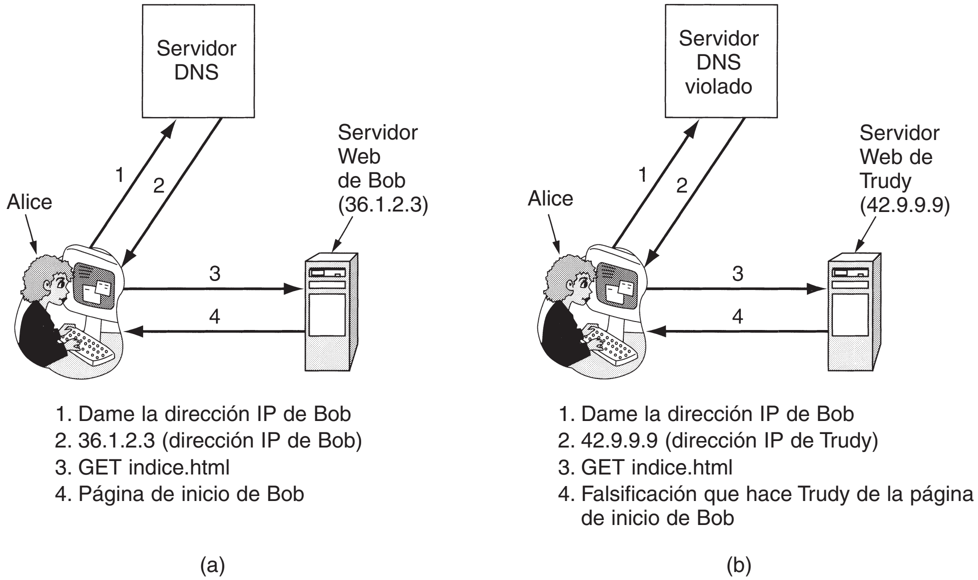


Figura 8-46. (a) Situación normal. (b) Un ataque basado en la violación de un DNS para modificar el registro de Bob.

¿Cómo engaña Trudy al DNS? Parece bastante fácil. Brevemente, Trudy puede engañar al servidor DNS en el ISP de Alice para que envíe una consulta a fin de encontrar la dirección de Bob. Desgraciadamente, puesto que el DNS utiliza UDP, el servidor DNS no puede verificar quién proporcionó la respuesta. Trudy puede explotar esta propiedad falsificando la respuesta esperada e introduciendo una dirección IP falsa en el caché del servidor DNS. Por simplicidad, supondremos que el ISP de Alice inicialmente no tiene una entrada para el sitio Web de Bob, *bob.com*. Si la tiene, Trudy puede esperar hasta que expire y probar otra vez (o utilizar otros trucos).

Trudy inicia el ataque enviando una solicitud de respuesta al ISP de Alice preguntando por la dirección IP de *bob.com*. Puesto que no hay entrada para este nombre DNS, el servidor de caché pide al servidor de nivel superior el dominio *com* para obtener uno. Sin embargo, Trudy evade al servidor *com* y regresa una respuesta falsa diciendo: “*bob.com* es 42.9.9.9”; sin embargo, esta dirección IP es la de ella. Si la respuesta falsa de Trudy llega primero al ISP de Alice, se almacenará en caché y la respuesta real se rechazará como respuesta no solicitada de una consulta que ya no está pendiente. Engañar a un servidor DNS para que instale una dirección IP falsa se conoce como **falsificación de DNS.** Un caché que contiene una dirección IP intencionalmente falsa como ésta se conoce como **caché envenenado.**

Por lo general, las cosas no son tan sencillas. Primero, el ISP de Alice verifica si la respuesta lleva la dirección de origen IP correcta del servidor de nivel superior. Pero debido a que Trudy puede colocar lo que quiera en ese campo IP, puede vencer esa prueba con facilidad puesto que las direcciones IP de los servidores de nivel superior tienen que ser públicas.

Segundo, para permitir que los servidores DNS digan cuál respuesta corresponde a cual solicitud, todas las solicitudes llevan un número de secuencia. Para falsificar el ISP de Alice, Trudy tiene que conocer su número de secuencia actual. La forma más fácil para que Trudy conozca el número de secuencia actual es que ella misma registre un dominio, digamos, *trudy-la-intrusa.com*. Supongamos que su dirección IP también es 42.9.9.9. Trudy también crea un servidor DNS para su nuevo dominio, *dns.trudy-la-intrusa.com*. El servidor DNS utiliza la dirección IP 42.9.9.9 de Trudy, puesto que ésta sólo tiene una computadora. Ahora Trudy tiene que informarle al ISP de Alice sobre su servidor DNS. Esto es fácil. Todo lo que tiene que hacer es pedir *foobar.trudy-la-intrusa.com* al IPS de Alice, lo que causará que dicho ISP pregunte al servidor *com* de nivel superior quién proporciona el nuevo dominio de Trudy.

Una vez que tiene seguro el *dns.trudy-la-intrusa.com* en el caché del ISP de Alice, el ataque real puede comenzar. Trudy ahora solicita *www.trudy-la-intrusa.com* al ISP de Alice. Como es natural, el ISP envía al servidor DNS de Trudy una consulta en la que se lo pide. Dicha consulta lleva el número de secuencia que Trudy está buscando. Rápidamente, Trudy pide al ISP de Alice que busque a Bob. Trudy responde de inmediato su propia pregunta enviando al ISP una respuesta falsificada, supuestamente del servidor *com* de nivel superior que dice: "*bob.com* es 42.9.9.9". Esta respuesta falsificada lleva un número de secuencia, un número más que el que acaba de recibir. Mientras esté allí, puede enviar una segunda falsificación con un número de secuencia incrementado en dos, y tal vez una docena más con números de secuencia crecientes. Uno de ellos está obligado a coincidir. El resto simplemente se elimina. Cuando llega la respuesta falsificada de Alice, se almacena en caché; cuando la respuesta real viene más tarde, se rechaza puesto que ninguna consulta está pendiente.

Cuando Alice busca a *bob.com*, se le indica que utilice 42.9.9.9, la dirección de Trudy. Ésta ha realizado un ataque hombre en medio desde la comodidad de su casa. En la figura 8-47 se ilustran los diversos pasos para este ataque. Para empeorar las cosas, ésta no es la única forma de falsificar un DNS. Hay muchas otras formas.

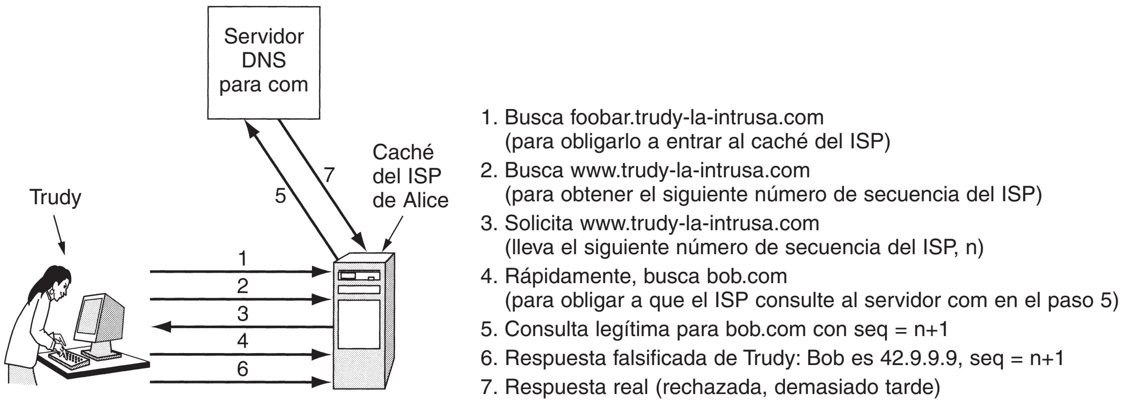


Figura 8-47. La forma en que Trudy falsifica el ISP de Alice.

DNS seguro

Este ataque específico puede frustrarse al hacer que los servidores DNS utilicen IDs aleatorios en cualquiera de las consultas en lugar de sólo contar, pero parece que cada vez que se tapa un hoyo, otro se destapa. El problema real es que el DNS se diseñó en la época en la que Internet era una herramienta de investigación para algunos cientos de universidades y ni Alice, ni Bob, ni mucho menos Trudy fueron invitados a la fiesta. En ese entonces la seguridad no era un problema; hacer que Internet funcionara era el problema. Con los años, el entorno ha cambiado radicalmente, por lo que en 1994 el IETF estableció un grupo funcional para hacer que DNS fuera seguro. Este proyecto se conoce como **DNSsec (seguridad DNS);** su salida se presenta en el RFC 2535. Desgraciadamente, DNSsec aún no se ha distribuido por completo, por lo que numerosos servidores DNS aún son vulnerables a ataques de falsificación.

DNSsec es en concepto muy sencillo. Se basa en la criptografía de clave pública. Cada zona DNS (en el sentido de la figura 7-4) tiene un par de claves pública/privada. Toda la información enviada por un servidor DNS se firma con la clave privada de la zona originaria, por lo que el receptor puede verificar su autenticidad.

DNSsec ofrece tres servicios fundamentales:

1. Prueba de en dónde se originaron los datos.
2. Distribución de la clave pública.
3. Autenticación de transacciones y solicitudes.

El servicio principal es el que se lista primero, el cual verifica que los datos que se están regresando ha sido probado por el dueño de la zona. El segundo es útil para almacenar y recuperar de manera segura claves públicas. El tercero es necesario para proteger contra ataques de repetición y falsificación. Observe que la confidencialidad no es un servicio ofrecido puesto que toda la información en el DNS se considera como pública. Puesto que la planificación en etapas del DNSsec se lleva algunos años, es esencial la capacidad que tienen los servidores concientes de la seguridad para interactuar con los servidores que no están conscientes de ella, lo cual implica que el protocolo no puede cambiarse. Veamos ahora algunos de los detalles.

Los registros DNS están agrupados en conjuntos llamados **RRSets** (**Conjuntos de Registro de Recursos**), en los que todos los registros tienen el mismo nombre, clase y tipo en un conjunto. Un RRSet puede contener múltiples registros *A*; por ejemplo, si un nombre DNS se resuelve en una dirección IP primaria y una secundaria. Los RRSets se extienden con varios nuevos tipos de registro (que se analizan a continuación). A cada RRSet se le aplica un *hash* de manera criptográfica (por ejemplo, utilizando MD5 o SHA-1). La clave privada de la zona firma (por ejemplo, utilizando RSA) el *hash*. La unidad de transmisión a clientes es el RRSet firmado. Al momento de la recepción de un RRSet firmado, el cliente puede verificar si fue firmado por la clave privada de la zona originaria. Si la firma concuerda, se aceptan los datos. Puesto que cada RRSet contiene su propia firma, los RRSets pueden cambiarse en cualquier lugar, incluso en servidores no confiables, sin poner en peligro la seguridad.

DNSsec introduce varios tipos nuevos de registros. El primero de éstos es el registro *KEY*. Estos registros mantienen la clave pública de una zona, usuario, *host* u otro personaje principal, el algoritmo criptográfico utilizado para firmar, el protocolo utilizado para transmisión y otros bits. La clave pública se almacena tal como está. Los certificados X.509 no se utilizan debido a su volumen. El campo de algoritmo contiene un 1 para las firmas MD5/RSA (la opción preferida) y otros valores para otras combinaciones. El campo de protocolo puede indicar el uso de IPsec y otros protocolos de seguridad, si es que hay.

SIG es el segundo nuevo tipo de registro. Contiene el *hash* firmado de acuerdo con el algoritmo especificado en el registro *KEY.* La firma se aplica a todos los registros del RRSet, incluyendo a cualquiera de los registros *KEY* presentes, pero excluyéndose a sí mismo. También contiene la fecha en la que la firma comienza su periodo de validación y cuando ésta expira, así como el nombre de quien firma y algunos otros elementos.

El diseño de DNSsec tiene la característica de que es posible mantener sin conexión una clave privada de una zona. Una o dos veces al día, el contenido de una base de datos de una zona puede transportarse de manera manual (por ejemplo, en CD-ROM) a una máquina desconectada en la que se localiza una clave privada. Todos los RRSets pueden firmarse ahí y los registros *SIG* producidos de esa manera pueden transportarse al servidor primario de la zona en CD-ROM. De esta manera, la clave privada puede almacenarse en un CD-ROM guardado en forma segura excepto cuando se inserta en la máquina desconectada para firmar los nuevos RRSets del día. Después de que se termina el proceso de firma, todas las copias de la clave se eliminan de la memoria y el disco y el CD-ROM son guardados. Este procedimiento reduce la seguridad electrónica a seguridad física, algo que las personas saben cómo tratar.

Este método de prefirmar los RRSets aumenta la velocidad de manera considerable el proceso de responder consultas puesto que no es necesario realizar criptografía sobre la marcha. La desventaja consiste en que se necesita una gran cantidad de espacio en disco para almacenar todas las claves y firmas en las bases de datos DNS. Algunos registros incrementarán diez veces el tamaño debido a la firma.

Cuando un proceso de cliente obtenga un RRSet firmado, debe aplicar la clave pública de la zona originaria para desencriptar el *hash*, calcular el *hash* mismo y comparar los dos valores. Si concuerdan, los datos se consideran válidos. Sin embargo, este procedimiento asume la manera de cómo obtiene el cliente la clave pública de la zona. Una forma es adquirirla de un servidor confiable, utilizando una conexión segura (por ejemplo, utilizando IPsec).

Sin embargo, en la práctica, se espera que los clientes serán preconfigurados con las claves públicas de todos los dominios de nivel superior. Si Alice ahora desea visitar el sitio Web de Bob, puede pedir al DNS el RRSet de *bob.com*, que contendrá su dirección IP y un registro *KEY* que contiene la clave pública de Bob. Este RRSet será firmado por el dominio *com* de nivel superior, por lo que Alice puede verificar fácilmente su validez. Un ejemplo de lo que podría contener este RRSet se muestra en la figura 8-48.

Una vez que Alice tiene una copia verificada de la clave pública de Bob, puede pedir al servidor DNS de Bob la dirección IP de *www.bob.com*. La clave privada de Bob firmará este RRSet, a fin de que Alice pueda verificar la firma del RRSet que Bob regresa. Si Trudy hace algo para inyectar un RRSet falso en cualquiera de los cachés, Alice puede detectar fácilmente su falta de autenticidad porque el registro *SIG* contenido será incorrecto.

Nombre del dominio	Tiempo de vida	Clase	Tipo	Valor
bob.com.	86400	IN	A	36.1.2.3
bob.com.	86400	IN	KEY	3682793A7B73F731029CE2737D...
bob.com.	86400	IN	SIG	86947503A8B848F5272E53930C...

Figura 8-48. Un RRSet de ejemplo para *bob.com*. El registro *KEY* es la clave pública de Bob. El registro *SIG* es el *hash* firmado del servidor *com* de nivel superior de los registros *A* y *KEY* para verificar su autenticidad.

Sin embargo, DNSsec también proporciona un mecanismo criptográfico para enlazar una respuesta a una consulta específica, a fin de evitar el tipo de falsificación utilizado por Trudy en la figura 8-47. Esta medida (opcional) de antifalsificación agrega a la respuesta un *hash* del mensaje de consulta firmado con la clave privada del receptor. Puesto que Trudy no conoce la clave privada del servidor *com* de nivel superior, no puede falsificar una respuesta a una consulta que el ISP de Alice haya enviado a dicho servidor. Ciertamente Trudy puede conseguir que su respuesta regrese primero, pero será rechazada debido a la firma inválida sobre la consulta con *hash*.

DNSsec también soporta algunos otros tipos de registros. Por ejemplo, el registro *CERT* puede utilizarse para almacenar certificados (por ejemplo, X.509). Este registro ha sido proporcionado porque algunas personas desean cambiar un DNS en una PKI. Si esto realmente sucede, está por verse. Detendremos nuestro análisis de DNSsec aquí. Para mayores detalles, por favor consulte el RFC 2535.

Nombres autocertificables

El DNS seguro no es la única posibilidad para asegurar los nombres. Un método completamente diferente se utiliza en el **Sistema de Archivos Seguro** (Mazières y cols., 1999). En este proyecto, los autores diseñaron un sistema de archivos seguro y escalable a nivel mundial, sin modificar el DNS (estándar) y sin utilizar certificados o suponer la existencia de una PKI. En esta sección mostraremos la forma en que estas ideas pueden aplicarse a Web. De manera acorde, en la descripción que daremos a continuación, utilizaremos la terminología de Web en lugar de la del sistema de archivos. Pero para evitar cualquier posible confusión, mientras que este esquema *podría* aplicarse a Web a fin de lograr una máxima seguridad, actualmente no está en uso y podría requerir cambios sustanciales de software para introducirla.

Comenzaremos suponiendo que cada servidor Web tiene un par de claves pública/privada. La esencia de la idea es que cada URL contiene un *hash* criptográfico del nombre del servidor y una clave pública como parte del URL. Por ejemplo, en la figura 8-49 podemos ver el URL para la foto de Bob. Comienza con el esquema HTTP usual seguido por el nombre DNS del servidor (*www.bob.com*). A continuación se encuentra un punto y coma y un *hash* de 32 caracteres. Al final está el nombre del archivo, nuevamente como es usual. Excepto por el *hash*, éste es un URL estándar. Con el *hash*, es un **URL autocertificable.**

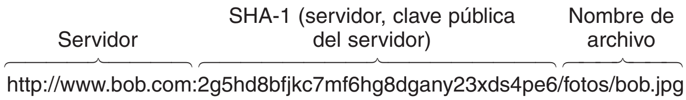


Figura 8-49. Un URL autocertificable que contiene un *hash* del nombre y la clave pública del servidor.

La pregunta obvia es: ¿Para qué es el *hash*? Éste se calcula concatenando el nombre DNS del servidor con la clave pública del servidor y ejecutando el resultado mediante la función SHA-1 para obtener un *hash* de 160 bits. En este esquema, el *hash* se representa como una secuencia de 32 dígitos y letras en minúsculas, a excepción de las letras “l” y “o” y los dígitos “1” y “0”, para evitar la confusión. Esto deja 32 dígitos y letras. Con 32 caracteres disponibles, cada uno puede codificar una cadena de 5 bits. Una cadena de 32 caracteres puede contener el *hash* SHA-1 de 160 bits. Realmente, no es necesario utilizar un *hash*; puede utilizarse la clave misma. La ventaja del *hash* es que se reduce la longitud del nombre.

La forma más sencilla (pero menos conveniente) para ver la foto de Bob, es que Alice simplemente teclee la cadena de la figura 8-49 en su navegador. Éste envía un mensaje al sitio Web de Bob pidiéndole la clave pública. Cuando llega la clave pública de Bob, el navegador concatena el nombre del servidor y la clave pública y ejecuta el algoritmo de *hash*. Si el resultado concuerda con el *hash* de 32 caracteres en el URL seguro, el navegador está seguro de que tiene la clave pública de Bob. Después de todo, debido a las propiedades de SHA-1, incluso aunque Trudy intercepte la solicitud y falsifique la respuesta, no tendrá manera de encontrar una clave pública que dé el *hash* esperado. Por lo tanto, se detectará cualquier interferencia por parte de Trudy. La clave pública de Bob puede almacenarse en caché para uso futuro.

Ahora Alice tiene que verificar que Bob tiene la clave privada correspondiente. Ella construye un mensaje que contiene una clave de sesión AES propuesta, una marca aleatoria y una marca de tiempo. A continuación encripta el mensaje con la clave pública de Bob y se la envía. Puesto que sólo Bob tiene la clave privada correspondiente, únicamente él puede desencriptar el mensaje y regresar la marca aleatoria encriptada con la clave AES. Una vez que recibe la marca aleatoria AES encriptada correcta, Alice sabe que está hablando con Bob. Además, Alice y Bob ahora tienen una clave de sesión AES para las solicitudes y respuestas *GET* subsecuentes.

Una vez que Alice tiene la foto de Bob (o cualquier página Web), puede marcarla, por lo que ya no tiene que teclear nuevamente todo el URL. Además, los URLs incrustados en páginas Web también pueden ser autocertificables, por lo que pueden utilizarse con sólo hacer clic en ellos, pero con la seguridad adicional de que sabe que la página regresada es la correcta. Otras formas de evitar los URLs autocertificables son obtenerlos a través de una conexión segura a un servidor confiable o tenerlos presentes en certificados X.509 firmados por CAs.

Otra forma de obtener URLs autocertificables podría ser conectarse con una máquina de búsqueda confiable tecleando su URL autocertificable (la primera vez) e ir a través del mismo protocolo como se describió anteriormente, lo que produciría una conexión autenticada segura con la máquina de búsqueda confiable. A continuación podría consultarse dicha máquina de búsqueda, y los resultados aparecerían en una página firmada llena de URLs autocertificables en los que se podría hacer clic sin tener que teclear cadenas largas.

Ahora veamos qué efectos tiene este método en la falsificación de DNS de Trudy. Si Trudy se las arregla para envenenar el caché del ISP de Alice, la solicitud de ésta podría ser falsamente entregada a Trudy en lugar de a Bob. Pero el protocolo ahora requiere que el destinatario de un mensaje original (es decir, Trudy) regrese una clave pública que produce el *hash* correcto. Si Trudy regresa su propia clave pública, Alice lo detectará inmediatamente porque el *hash* SHA-1 no coincidirá con el URL autocertificable. Si Trudy regresa la clave pública de Bob, Alice no detectará el ataque, pero encriptará su siguiente mensaje, utilizando la clave de Bob. Trudy obtendrá el mensaje, pero no tendrá forma de desencriptarlo para extraer la clave AES y la marca aleatoria. De cualquier forma, todo lo que la falsificación DNS puede hacer es proporcionar un ataque de negación de servicio.

8.9.3 SSL—La Capa de Sockets Seguros

La asignación de nombres segura es un buen inicio, pero hay mucho más sobre la seguridad en Web. El siguiente paso son las conexiones seguras. A continuación veremos cómo pueden lograrse las conexiones seguras.

Cuando Web irrumpió a la vista pública, inicialmente sólo se utilizó para distribuir páginas estáticas. Sin embargo, pronto algunas compañías tuvieron la idea de utilizarla para transacciones financieras, como para comprar mercancía con tarjeta de crédito, operaciones bancarias en línea y comercio electrónico de acciones. Estas aplicaciones crearon una demanda de conexiones seguras. En 1995, Netscape Communications Corp, el entonces fabricante líder de navegadores, respondió a esta demanda con un paquete de seguridad llamado **SSL** (**Capa de Sockets Seguros**). En la actualidad, este software y su protocolo se utilizan ampliamente, incluso por Internet Explorer, por lo que vale la pena examinarlo en detalle.

SSL construye una conexión segura entre los dos sockets, incluyendo

1. Negociación de parámetros entre el cliente y el servidor.
2. Autenticación tanto del cliente como del servidor.
3. Comunicación secreta.
4. Protección de la integridad de los datos.

Ya hemos visto antes estos elementos, por lo que no hay necesidad de volver a tratarlos.

En la figura 8-50 se ilustra la posición de SSL en la pila de protocolos usuales. Efectivamente, es una nueva capa colocada entre la capa de aplicación y la de transporte, que acepta solicitudes del navegador y enviándolas al TCP para transmitir al servidor. Una vez que se ha establecido la conexión segura, el trabajo principal de SSL es manejar la compresión y encriptación. Cuando HTTP se utiliza encima de SSL, se conoce como **HTTPS** (**HTTP Seguro**), aunque es el protocolo HTTP estándar. Sin embargo, algunas veces está disponible en un nuevo puerto (443) en lugar de en uno estándar (80). Además, SSL no está restringido a utilizarse sólo con navegadores Web, pero ésa es la aplicación más común.

El protocolo SSL ha pasado por varias versiones. A continuación sólo trataremos la versión 3, que es la versión más ampliamente utilizada. SSL soporta una variedad de algoritmos y opciones

Aplicación (HTTP)
Seguridad (SSL)
Transporte (TCP)
Red (IP)
Enlace de datos (PPP)
Física (módem, ADSL, TV por cable)

Figura 8-50. Capas (y protocolos) para una navegación de usuario doméstico con SSL.

diferentes. Entre dichas opciones se incluyen la presencia o ausencia de compresión, los algoritmos criptográficos a utilizar y algunos asuntos relacionados con la exportación de restricciones en la criptografía. La última es la destinada principalmente para asegurarse de que se utilice criptografía seria sólo cuando ambos lados de la conexión estén en los Estados Unidos. En otros casos, las claves se limitan a 40 bits, lo que los criptógrafos consideran como una broma. El gobierno de los Estados Unidos obligó a Netscape a incluir esta restricción para obtener una licencia de exportación.

SSL consiste en dos subprotocolos, uno para establecer una conexión segura y otro para utilizarla. Comencemos viendo cómo se establecen las conexiones seguras. En la figura 8-51 se muestra el subprotocolo de establecimiento de conexión. Comienza con el mensaje 1, cuando Alice envía una solicitud a Bob para que establezca una conexión. La solicitud especifica la versión SSL que tiene Alice y sus preferencias con respecto a los algoritmos criptográficos y de compresión. También contiene una marca aleatoria, R_A, para utilizarse más tarde.

Ahora es el turno de Bob. En el mensaje 2, Bob realiza una elección de entre los diversos algoritmos que Alice puede soportar y envía su propia marca aleatoria, R_B. Después, en el mensaje 3, envía un certificado que contiene su clave pública. Si este certificado no está firmado por alguna autoridad bien conocida, también envía una cadena de certificados que pueden seguirse hasta encontrar uno. Todos los navegadores, incluyendo el de Alice, vienen precargados con aproximadamente 100 claves públicas, por lo que si Bob puede establecer una cadena anclada a una de ellas, Alice podrá verificar la clave pública de Bob. En este punto Bob podría enviar algunos otros mensajes (por ejemplo, podría solicitar el certificado de la clave pública de Alice). Cuando Bob termina, envía el mensaje 4 para indicar a Alice que es su turno.

Alice responde eligiendo una **clave premaestra** aleatoria de 384 bits y enviándola a Bob encriptada con la clave pública de él (mensaje 5). La clave de sesión real utilizada para encriptar datos se deriva de la clave premaestra combinada con las dos marcas aleatorias en una forma compleja. Después de que se ha recibido el mensaje 5, tanto Alice como Bob pueden calcular la clave de sesión. Por esta razón, Alice indica a Bob que cambie al nuevo cifrado (mensaje 6) y también que ha terminado con el establecimiento del subprotocolo (mensaje 7). Después Bob confirma que ha recibido la indicación (mensajes 8 y 9).

Sin embargo, aunque Alice sabe quién es Bob, éste no sabe quién es Alice (a menos que Alice tenga una clave pública y un certificado correspondiente para ella, una situación no probable para un individuo). Por lo tanto, el primer mensaje de Bob puede ser una solicitud para que Alice inicie una sesión utilizando un nombre de inicio de sesión y una contraseña previamente establecidos.

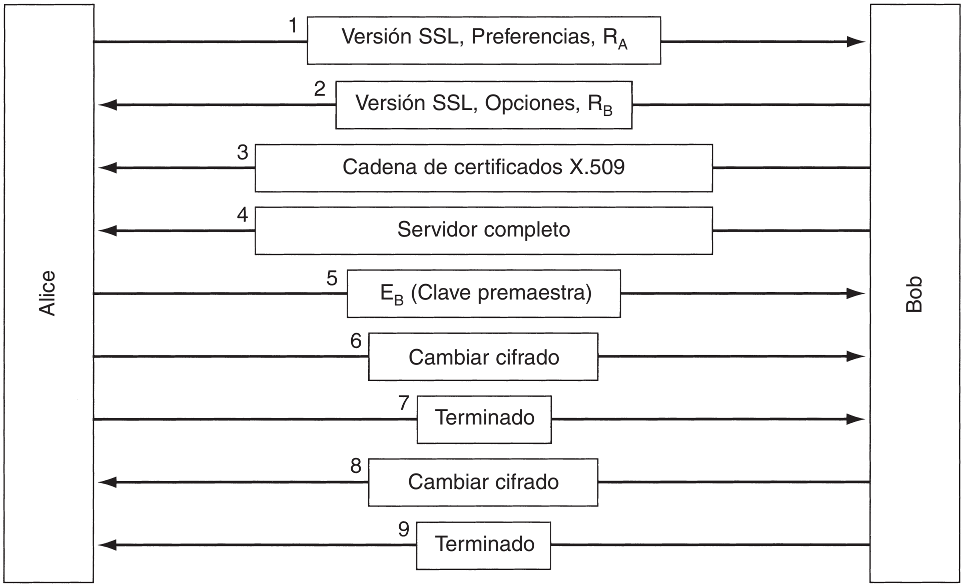


Figura 8-51. Una versión simplificada del subprotocolo de establecimiento de conexión SSL.

Sin embargo, el protocolo de inicio de sesión está fuera del alcance de SSL. Una vez que se ha consumado, por cualquier medio, puede comenzar el transporte de los datos.

Como se mencionó anteriormente, SSL soporta múltiples algoritmos criptográficos. El más robusto utiliza triple DES con tres claves separadas para encriptación y SHA-1 para la integridad de mensajes. Esta combinación es relativamente lenta, por lo que se utiliza principalmente para operaciones bancarias y otras aplicaciones en las que se requiere la seguridad de mayor nivel. Para las aplicaciones comunes de comercio electrónico, se utiliza RC4 con una clave de 128 bits para encriptación y MD5 se utiliza para la autenticación de mensajes. RC4 toma la clave de 128 bits como semilla y la expande a un número mucho más grande para uso interno. Después utiliza este número interno para generar un flujo de claves. A éste se le aplica un OR exclusivo con el texto llano para proporcionar un cifrado clásico de flujo, como vimos en la figura 8-14. Las versiones de exportación también utilizan RC4 con claves de 128 bits, 88 de los cuales se hacen públicos para que el cifrado sea fácil de romper.

Para un transporte real se utiliza un segundo subprotocolo, como se muestra en la figura 8-52. Los mensajes que provengan del navegador primero se dividen en unidades de hasta 16 KB. Si se activa la compresión, cada unidad se comprime por separado. Después de eso, se deriva una clave secreta a partir de las dos marcas aleatorias y la clave premaestra se concatena con el texto comprimido y al resultado se le aplica un *hash* con el algoritmo de *hash* acordado (por lo general MD5). Este *hash* se agrega a cada fragmento como el MAC. Después, el fragmento comprimido y el MAC se encriptan con el algoritmo de encriptación simétrico acordado (por lo general, aplicándole un OR exclusivo con el flujo de claves RC4). Por último, se agrega un encabezado de fragmento y el fragmento se transmite a través de la conexión TCP.

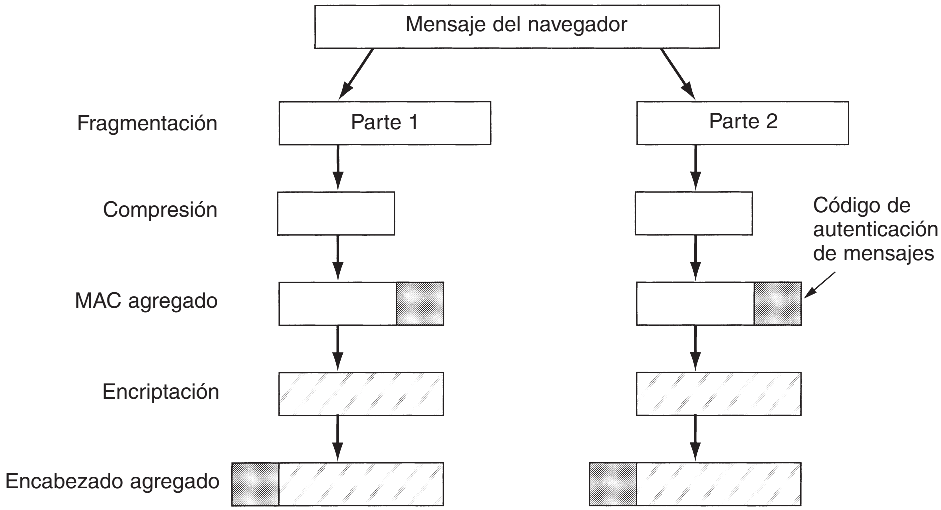


Figura 8-52. Transmisión de datos mediante SSL.

Sin embargo, es necesaria una advertencia. Puesto que se ha mostrado que el RC4 tiene claves débiles que pueden criptoanalizarse con facilidad, la seguridad de SSL mediante RC4 no es muy confiable (Fluhrer y cols., 2001). Los navegadores que permiten que el usuario elija el conjunto de cifrado deben configurarse para utilizar todo el tiempo triple DES con claves de 168 bits y SHA-1, aunque esta combinación es más lenta que RC4 y MD5.

Otro problema con SSL es que tal vez los personajes principales no tienen certificados e incluso si los tienen, no siempre verifican que coincidan las claves que se utilizan.

En 1996, Netscape Communications Corp. mandó el SSL a la IETF para su estandarización. El resultado fue **TLS** (**Seguridad de Capa de Transporte**). Se describe en el RFC 2246.

Los cambios hechos a SSL fueron relativamente pequeños, pero sólo lo suficiente para que SSL versión 3 y TLS no puedan interoperar. Por ejemplo, la forma en que se deriva una clave de sesión a partir de la clave premaestra y las marcas aleatorias se cambió para hacer que la clave fuera más fuerte (es decir, fuera difícil de criptoanalizar). La versión TLS también se conoce como SSL versión 3.1. Las primeras implementaciones aparecieron en 1999, pero aún no es claro si TLS reemplazará a SSL en la práctica, aunque es ligeramente más fuerte. Sin embargo, permanece el problema con las claves débiles RC4.

8.9.4 Seguridad de código móvil

La asignación de nombres y las conexiones son dos áreas de preocupación que se relacionan con la seguridad en Web. Pero hay más. En los primeros días, cuando las páginas Web eran simplemente archivos HTML estáticos, no contenían código ejecutable. Ahora con frecuencia contienen pequeños programas, incluyendo subprogramas de Java, controles ActiveX y JavaScripts. Descargar y ejecutar tal **código móvil** obviamente es un riesgo de seguridad masivo, por lo que se han

diseñado varios métodos para minimizarlo. A continuación daremos un vistazo a algunos de los problemas surgidos debido al código móvil y a algunos métodos para tratarlos.

Seguridad de subprogramas de Java

Los subprogramas de Java son pequeños programas de Java compilados a un lenguaje de máquina orientado a pilas llamado **JVM** (**Máquina Virtual de Java**). Pueden colocarse en una página Web para descargarlos junto con dicha página. Después de que la página se carga, los subprogramas se insertan en un intérprete JVM dentro del navegador, como se ilustra en la figura 8-53.

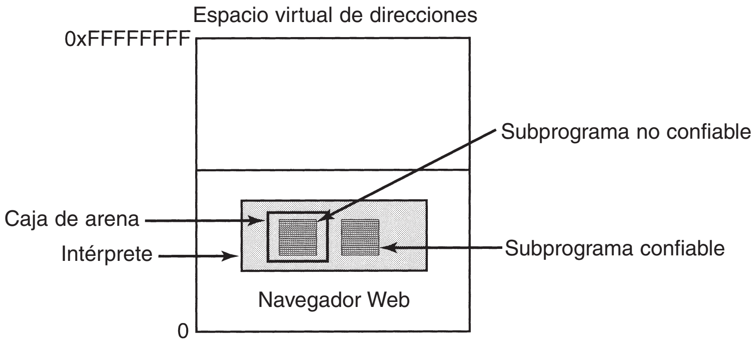


Figura 8-53. Los subprogramas pueden ser interpretados por un navegador Web.

La ventaja de ejecutar código interpretado en lugar de compilado es que cada instrucción es examinada por el intérprete antes de ser ejecutada. Esto da al intérprete la oportunidad de verificar si la dirección de la instrucción es válida. Además, las llamadas de sistema también son atrapadas e interpretadas. La forma en que se manejan estas llamadas depende de la política de seguridad. Por ejemplo, si un subprograma es confiable (por ejemplo, proviene del disco local), sus llamadas de sistema podrían transportarse sin pregunta alguna. Sin embargo, si un subprograma no es confiable (por ejemplo, proviene de Internet), podría encapsularse en lo que se conoce como **caja de arena** para restringir su comportamiento y atrapar sus intentos por utilizar los recursos del sistema.

Cuando un subprograma intenta utilizar un recurso del sistema, su llamada se pasa al monitor de seguridad para que la apruebe. El monitor examina la llamada a la luz de la política de seguridad local y después toma una decisión para permitirla o rechazarla. De esta manera, es posible proporcionar acceso a los subprogramas a algunos recursos pero no a todos. Desgraciadamente, la realidad es que el modelo de seguridad no funciona del todo bien y con frecuencia surgen errores.

ActiveX

Los controles ActiveX son programas binarios Pentium que pueden incrustarse en páginas Web. Cuando se encuentra alguno de ellos, se realiza una verificación para ver si se debe ejecutar, y si pasa la prueba, se ejecuta. No está interpretado o puesto en una caja de arena de ninguna

manera, por lo que tiene tanto poder que cualquier otro programa de usuario, y puede hacer mucho daño. Por lo tanto, toda la seguridad recae en la decisión de si se ejecuta o no el control ActiveX.

El método que Microsoft eligió para tomar esta decisión se basa en la idea de la **firma de código.** Cada control ActiveX está acompañado por una firma digital —un *hash* del código que está firmado por su creador utilizando la encriptación de clave pública. Cuando aparece un control ActiveX, el navegador primero verifica la firma para asegurarse de que no ha sido alterada en el camino. Si la firma es correcta, el navegador verifica a continuación sus tablas internas para ver si el creador del programa es confiable o hay una cadena de confianza hacia un creador confiable. Si el creador es confiable, el programa se ejecuta; de lo contrario, no se ejecuta. El sistema de Microsoft para verificar controles ActiveX se conoce como **Authenticode.**

Es útil comparar los métodos de Java y ActiveX. Con el método de Java, no se realiza ningún intento por determinar quién escribió el subprograma. En su lugar, un intérprete en tiempo de ejecución se asegura de que el subprograma no haga lo que el dueño de la máquina haya prohibido. En contraste, con la firma de código, no hay intento por supervisar el comportamiento en tiempo de ejecución del código móvil. Si proviene de un origen confiable y no ha sido modificado en el camino, simplemente se ejecuta. No se realiza ningún intento por ver si el código es maligno o no. Si el programador original *destinó* el código para formatear el disco duro y después borra la *flash* ROM a fin de que la computadora nunca vuelva a encenderse, y si el programador ha sido certificado como confiable, el código se ejecutará y destruirá la computadora (a menos que los controles ActiveX hayan sido desactivados en el navegador).

Muchas personas piensan que confiar en software de una compañía desconocida es temible. Para demostrar el problema, un programador de Seattle formó una compañía de software y la certificó como confiable, lo cual es fácil de hacer. Después escribió un control ActiveX que apagaba limpiamente la máquina y lo distribuyó ampliamente. Apagó muchas máquinas, pero podían encenderse nuevamente, por lo que no causó ningún daño. Simplemente trataba de exponer el problema al mundo. La respuesta oficial fue revocar el certificado a este control ActiveX específico, lo cual acabó con un corto episodio vergonzoso, pero el problema subyacente aún está allí para que un programador con no buenas intenciones se aproveche de él (Garfinkel y Spafford, 2002). Puesto que no hay forma de vigilar miles de compañías de software que podrían escribir código móvil, la técnica de firma de código es un desastre al acecho.

JavaScript

JavaScript no tiene ningún modelo de seguridad formal, pero tiene una larga historia de implementaciones defectuosas. Cada fabricante maneja la seguridad de forma diferente. Por ejemplo, la versión 2 de Netscape Navigator utilizó algo semejante al modelo de Java, pero en la versión 4 se abandonó por un modelo de firma de código.

El problema fundamental es que permitir la ejecución de código extraño en su máquina es abrir la puerta a los problemas. Desde el punto de vista de la seguridad, es como invitar a un ladrón a su casa y después tratar de vigilarlo cuidadosamente de manera que no pueda ir de la cocina

a la sala. Si algo inesperado sucede y está distraído por un momento, pueden suceder cosas extrañas. El suspenso aquí es que el código móvil permite gráficos vistosos e interacción rápida, y muchos diseñadores de sitios Web piensan que esto es más importante que la seguridad, especialmente cuando es la máquina de alguien más la que está en riesgo.

Virus

Los virus son otra forma de código móvil. Sólo que a diferencia de los ejemplos anteriores, los virus no se invitan de ninguna manera. La diferencia entre un virus y el código móvil ordinario es que los virus se escriben para que se reproduzcan a sí mismos. Cuando llegan los virus, ya sea por medio de una página Web, un archivo adjunto de correo electrónico, o algún otro medio, por lo general inician infectando los programas ejecutables del disco. Cuando se ejecuta alguno de estos programas, el control se transfiere al virus, que por lo general trata de esparcirse a sí mismo a otras máquinas, por ejemplo, enviando por correo electrónico copias de sí mismo a cualquier persona que se encuentre en la libreta de direcciones de la víctima. Algunos virus infectan el sector de arranque del disco duro, por lo que cuando la máquina se inicia, el virus se ejecuta. Los virus se han convertido en un gran problema en Internet y han causado que se pierdan miles de millones de dólares por los daños. No hay solución a este problema. Tal vez podría ayudar una nueva generación de sistemas operativos basados en *microkernels* seguros y el compartimiento de usuarios, procesos y recursos.

8.10 ASPECTOS SOCIALES

Internet y su tecnología de seguridad es un área donde confluyen los aspectos sociales, las políticas de seguridad y la tecnología, frecuentemente con consecuencias importantes. A continuación sólo examinaremos con brevedad tres áreas: privacidad, libertad de expresión y derechos de autor. No es necesario decir que sólo echaremos un vistazo. Para mayor información, vea (Anderson, 2001; Garfinkel con Spafford, 2002, y Schneier, 2000). En Internet también puede encontrar mucho material. Simplemente introduzca en cualquier máquina de búsqueda palabras como “privacidad”, “censura” y “derechos de autor”. Además, vea el sitio Web de este libro para obtener algunos vínculos.

8.10.1 Privacidad

¿Las personas tienen derecho a la privacidad? Buena pregunta. La Cuarta Enmienda de los Estados Unidos prohíbe que el gobierno busque en las casas, documentos y bienes de las personas sin una razón válida y restringe las circunstancias en las que se deben emitir las órdenes de cateo. Por lo tanto, la privacidad ha estado en la agenda pública aproximadamente durante 200 años, por lo menos en Estados Unidos.

En la década pasada cambió la facilidad con la que el gobierno puede espiar a sus ciudadanos y la facilidad con la que éstos pueden evitar tal espionaje. En el siglo XVIII, para que el gobierno pudiera buscar en los documentos de los ciudadanos, tenía que enviar un policía a caballo a la granja de dicho ciudadano exigiendo ver ciertos documentos. Era un procedimiento engorroso. Hoy en día, las compañías telefónicas y los proveedores de Internet proporcionan con facilidad intervenciones telefónicas cuando se les presenta una orden de cateo. Esto facilita la vida del policía y ya no hay peligro de que se caiga del caballo.

La criptografía cambia todo esto. Cualquiera que se tome la molestia de bajar e instalar PGP y quien utilice una clave bien custodiada de fuerza alien puede estar seguro de que nadie en el universo conocido puede leer su correo electrónico, haya o no orden de cateo. Los gobiernos entienden bien esto y no les gusta. La privacidad real para ellos significa que les será mucho más difícil espiar a los criminales de todo tipo, y todavía les será más difícil espiar a los reporteros y oponentes políticos. En consecuencia, algunos gobiernos restringen o prohíben el uso o exportación de la criptografía. Por ejemplo, en Francia, antes de 1999, la criptografía estaba prohibida a menos que se le proporcionaran las claves al gobierno.

Esto no sólo sucedía en Francia. En abril de 1993, el gobierno de Estados Unidos anunció su intención de hacer que un criptoprocesador de hardware, el **procesador clipper,** fuera el estándar en toda la comunicación en red. De esta manera, se dijo, la privacidad de los ciudadanos estaría garantizada. También se mencionó que el procesador proporcionaría al gobierno la capacidad de desencriptar todo el tráfico a través de un esquema llamado **depósito de claves,** el cual permitía que el gobierno accediera a todas las claves. Sin embargo, se prometió que sólo se espiaría cuando se tuviera una orden de cateo válida. No es necesario decir que se generó una gran controversia por esta situación, en la que los activistas de la privacidad condenaban todo el plan y los oficiales del cumplimiento de la ley la aclamaban. En algún momento, el gobierno se retractó y abandonó la idea.

En el sitio Web de la Electronic Frontier Foundation, *www.eff.org*, está disponible una gran cantidad de información acerca de la privacidad electrónica.

Retransmisores de correo anónimos

PGP, SSL y otras tecnologías hacen posible que dos partes establezcan comunicación segura y autenticada, libre de vigilancia e interferencia de terceros. Sin embargo, algunas veces la privacidad se aplica mejor cuando *no* hay autenticación, es decir, haciendo que la comunicación sea anónima. El anonimato podría quererse para los mensajes punto a punto, grupos de noticias o ambos.

Veamos algunos ejemplos. Primero, los disidentes políticos que viven bajo regímenes autoritarios con frecuencia desean comunicarse de manera anónima para evitar ser encarcelados o asesinados. Segundo, por lo general las acciones ilegales en muchas organizaciones gubernamentales, educacionales, corporativas, entre otras, son denunciadas por personas que con frecuencia prefieren permanecer en el anonimato para evitar represalias. Tercero, las personas con creencias religiosas, políticas y sociales impopulares podrían querer comunicarse entre ellas a través de correo electrónico o grupos de noticias sin exponerse a sí mismos. Cuarto, las personas podrían desear discutir el alcoholismo, las enfermedades mentales, el acoso sexual, el abuso a infantes o ser

miembros de una minoría perseguida de un grupo de noticias sin revelar su identidad. Por supuesto, existen muchos otros ejemplos.

Consideremos un ejemplo específico. En la década de 1990 algunos críticos publicaron sus puntos de vista sobre un grupo religioso no tradicional, en un grupo de noticias de USENET a través de un **retransmisor de correo anónimo.** Este servidor permitía que los usuarios crearan pseudónimos y le enviaran correo electrónico, y después dicho servidor volvía a enviar o publicar tal correo utilizando el pseudónimo creado, de manera que nadie podía saber de dónde provenía realmente el mensaje. Algunas publicaciones revelaron información que el grupo religioso afirmaba eran secretos comerciales y documentos con derechos de autor. Por lo tanto, dicho grupo fue con las autoridades locales y les dijo que sus secretos comerciales habían sido revelados y que sus derechos de autor habían sido violados, lo cual eran delitos en el lugar donde se localizaba el servidor. En consecuencia, se produjo un juicio y el operador del servidor fue obligado a entregar la información de correspondencia, la cual reveló las verdaderas identidades de las personas quienes realizaron las publicaciones. (Incidentalmente, ésta no fue la primera vez que una religión no estaba de acuerdo con que alguien revelara sus secretos: William Tyndale fue quemado en la hoguera en 1536 por traducir al inglés la Biblia.)

Un segmento considerable de la comunidad de Internet se indignó por esta brecha de confidencialidad. La conclusión a la que todos llegaron es que no sirve de nada un retransmisor anónimo que almacena una correspondencia entre las direcciones reales de correo electrónico y los pseudónimos (llamado retransmisor de correo de tipo 1). Este caso estimuló a varias personas a diseñar retransmisores de correo anónimos que pudieran resistir ataques de citaciones legales.

Estos nuevos retransmisores, con frecuencia llamados **retransmisores de correo cypherpunks,** funcionan como se describe a continuación. El usuario produce un mensaje de correo electrónico, lleno de encabezados RFC 822 (excepto *From*:, por supuesto), lo encripta con la clave pública del retransmisor y lo envía a éste. Ahí se eliminan los encabezados externos RFC 822, el contenido se desencripta y el mensaje es retransmitido. El retransmisor no tiene cuentas ni mantiene registros, por lo que aunque el servidor se confisque posteriormente, no contiene ni una huella de los mensajes que han pasado a través de él.

Muchos usuarios que desean el anonimato encadenan sus solicitudes a través de múltiples retransmisores anónimos, como se muestra en la figura 8-54. Aquí, Alice desea enviar a Bob una tarjeta del Día de San Valentín en verdad anónima, por lo que utiliza tres retransmisores. Redacta el mensaje, M, y coloca en él un encabezado que contiene la dirección de correo electrónico de Bob. Después encripta todo el mensaje con la clave pública del retransmisor 3, E_3 (indicado por el sombreado horizontal). Para esto anexa al principio un encabezado con la dirección de correo electrónico en texto llano del retransmisor 3. En la figura, este mensaje es el que se muestra entre los retransmisores 2 y 3.

A continuación encripta este mensaje con la clave pública del retransmisor 2, *E2* (indicado por el sombreado vertical) y anexa al principio un encabezado en texto llano que contiene la dirección de correo electrónico del retransmisor 2. Este mensaje se muestra en la figura 8-54 entre los retransmisores 1 y 2. Por último, Alice encripta todo el mensaje con la clave pública del retransmisor 1, E_1, y anexa al inicio un encabezado en texto llano con la dirección de correo electrónico del retransmisor 1. En la figura, este mensaje es el que está a la derecha de Alice y también es el que ella transmite en realidad.

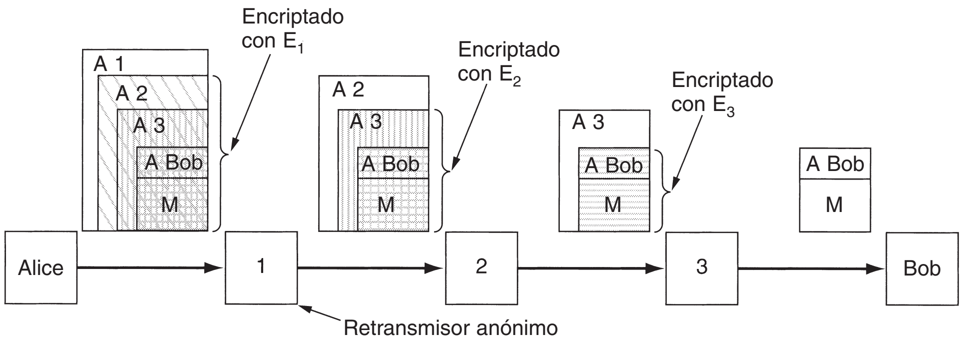


Figura 8-54. Cómo utiliza Alice tres retransmisores de correo para enviar un mensaje a Bob.

Cuando el mensaje llega al retransmisor de correo 1, el encabezado exterior se elimina. El cuerpo se desencripta y después se envía al retransmisor de correo 2. En los otros dos retransmisores se realizan pasos similares.

Aunque para cualquiera es muy difícil rastrear el mensaje final de regreso a Alice, muchos retransmisores de correo toman precauciones de seguridad adicionales. Por ejemplo, tal vez mantengan los mensajes por un tiempo aleatorio, agreguen o eliminen basura al final de un mensaje, y reordenen los mensajes, todo esto para dificultar que alguien pueda indicar cuál mensaje de un retransmisor de correo corresponde a qué entrada, a fin de frustrar el análisis de tráfico. Para una descripción de un sistema que representa lo último en correo electrónico anónimo, vea (Mazières y Kaashoek, 1998).

El anonimato no se limita al correo electrónico. También existen servicios que permiten la navegación anónima en Web. El usuario configura su navegador para utilizar el *anonymizer* como un *proxy*. De ahí en adelante, todas las solicitudes HTTP se dirigirán al anonymizer, el cual solicitará y regresará la página. El sitio Web ve al *anonymizer*, y no al usuario, como el origen de la solicitud. Siempre y cuando el *anonymizer* se abstenga de mantener una bitácora, nadie podrá determinar quién solicitó qué página.

8.10.2 Libertad de expresión

La privacidad se refiere a los individuos que desean restringir lo que otras personas ven en ellos. Un segundo problema social clave es la libertad de expresión, y su aspecto opuesto, la censura, que tiene que ver con el hecho de que los gobiernos desean restringir lo que los individuos pueden leer y publicar. Debido a que la Web contiene millones y millones de páginas, se ha vuelto un paraíso de censura. Dependiendo de la naturaleza e ideología del régimen, entre el material prohibido podrían encontrarse los sitios Web que contengan cualquiera de lo siguiente:

1. Material inapropiado para niños o adolescentes.
2. Odio dirigido a varios grupos religiosos, étnicos o sexuales, entre otros.
3. Información sobre democracia y valores democráticos.

4. Relatos de eventos históricos que contradigan la versión del gobierno.
5. Manuales para abrir candados, construir armas, encriptar mensajes, etcétera.

La respuesta común es prohibir los sitios malos.

Algunas veces los resultados son inesperados. Por ejemplo, algunas bibliotecas públicas han instalado filtros Web en sus computadoras para que sean aptas para los niños y bloqueados los sitios pornográficos. Los filtros vetan los sitios que se encuentran en sus listas negras y también verifican las páginas antes de desplegarlas para ver si contienen palabras obscenas. En Loudoun County, Virginia, sucedió que el filtro bloqueó la búsqueda que un cliente realizó para encontrar información sobre el cáncer de mama porque el filtro vio la palabra "mama". Dicho usuario de la biblioteca demandó al condado Loudoun. Sin embargo, en Livermore, California, después de que se sorprendió a un niño de 12 años de edad viendo pornografía, su padre demandó a la biblioteca por *no* instalar un filtro. ¿Qué tenía que hacer la biblioteca?

Mucha gente ha obviado el hecho de que World Wide Web es una red mundial. Cubre a todo el mundo. No todos los países están de acuerdo en lo que debe permitirse en Web. Por ejemplo, en noviembre de 2000, una corte de Francia ordenó a Yahoo, una corporación de California, que bloqueara a sus usuarios franceses para que no pudieran ver las subastas de objetos de recuerdo nazis, porque poseer tal material viola las leyes francesas. Yahoo apeló en una corte de Estados Unidos, la cual le dio la razón, pero aún está lejos de resolverse el problema de dónde aplicar las leyes de quién.

Simplemente imagínese. ¿Qué pasaría si alguna corte de Utah ordenara a Francia que bloqueara los sitios Web relacionados con el vino porque no cumplen con las muy estrictas leyes de Utah sobre el alcohol? Suponga que China demandara que todos los sitios Web que tienen que ver con la democracia fueran prohibidos porque no son del interés del Estado. ¿Las leyes iraníes sobre la religión se aplican a la Suecia más liberal? ¿Puede Arabia Saudita bloquear los sitios Web que tienen que ver con los derechos de la mujer? Todo el problema es un verdadera caja de Pandora.

Un comentario relevante de John Gilmore es: "la red interpreta la censura como una avería y encuentra una ruta alterna". Para una implementación concreta, considere el **servicio eternidad** (Anderson, 1996). Su objetivo es asegurarse de que la información publicada no pueda ser eliminada o reescrita, como era común en la Unión Soviética durante el reinado de Josef Stalin. Para utilizar el servicio eternidad, el usuario especifica cuánto tiempo se mantendrá el material, paga una cuota proporcional a su duración y tamaño, y lo carga. Después de eso, nadie puede eliminarlo o modificarlo, ni siquiera quien lo cargó.

¿Cómo se puede implementar tal servicio? El modelo más sencillo es utilizar un sistema de igual a igual en el que los documentos almacenados se coloquen en docenas de servidores participantes, cada uno de los cuales obtenga una parte de la cuota y, por lo tanto, un incentivo para unirse al sistema. Los servidores deben esparcirse a través de muchas jurisdicciones legales para obtener una máxima elasticidad. Las listas de los 10 servidores seleccionados al azar podrían almacenarse en forma segura en varios lugares, por lo que si algunos estuvieran en peligro, otros aún existirían. Una autoridad dispuesta a destruir el documento nunca estará segura de que ha encontrado todas las copias. El sistema también podría repararse a sí mismo; por ejemplo, si se sabe que se han destruido algunas copias, los sitios restantes podrían intentar encontrar nuevos depósitos para reemplazarlas.

El servicio eternidad fue la primera propuesta en lo que se refiere a sistemas anticensura. Desde entonces se han propuesto otros sistemas y, en algunos casos, se han implementado. Asimismo, se han agregado algunas nuevas características, como encriptación, anonimato y tolerancia a fallas. Con frecuencia los archivos se dividen en múltiples fragmentos, los cuales se almacenan en muchos servidores. Algunos de estos sistemas son Freenet (Clarke y cols., 2002), PASIS (Wylie y cols., 2000) y Publius (Waldman y cols., 2000). En (Serjantov, 2002), se reporta más trabajo.

En la actualidad, cada vez más países tratan de regular la exportación de valores intangibles, entre los que se encuentran sitios Web, software, documentos científicos, correo electrónico, servicios de ayuda telefónica, entre otros. Incluso en el Reino Unido, que tiene una tradición de siglos de libertad de expresión, ahora está considerando seriamente las leyes muy restrictivas, las cuales podrían, por ejemplo, definir las discusiones técnicas entre un profesor británico y su estudiante extranjero de la Universidad de Cambridge como exportación regulada que necesita una licencia del gobierno (Anderson, 2002). No es necesario decir que tales políticas son controversiales.

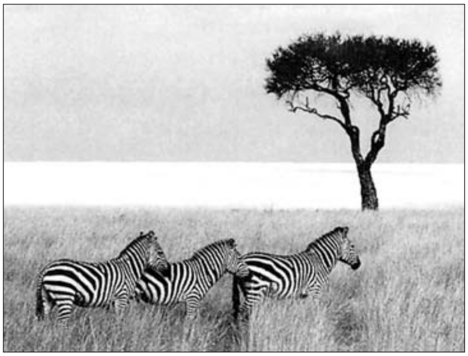
Esteganografía

En los países en donde abunda la censura, los disidentes con frecuencia tratan de utilizar la tecnología para evadirla. La criptografía permite el envío de mensajes secretos (aunque tal vez no legalmente), pero si el gobierno piensa que Alice es una Mala Persona, el simple hecho de que ella se esté comunicando con Bob podría ponerlo a él también en esta categoría, pues los gobiernos represivos entienden el concepto de clausura transitiva, aunque no entiendan bien las matemáticas. Los retransmisores de correo anónimos pueden ayudar, pero si están prohibidos domésticamente, y los mensajes dirigidos a extranjeros requieren una licencia de exportación por parte del gobierno, no serían de mucha ayuda. Pero Web sí puede.

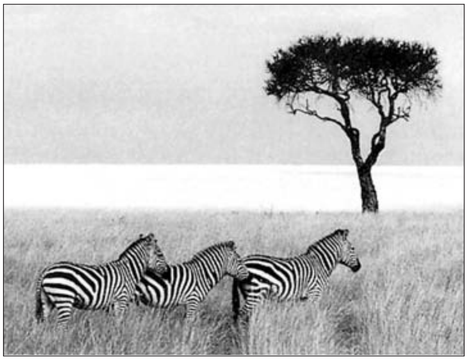
Las personas que desean comunicarse de manera secreta con frecuencia tratan de ocultar el hecho de que se está realizando la comunicación. La ciencia de ocultar mensajes se conoce como **esteganografía,** cuyo origen proviene de las palabras griegas correspondientes a “escritura encubierta”. De hecho, los antiguos griegos la utilizaron. Herodoto escribió sobre un general que rapó a un mensajero, tatuó un mensaje en el cuero cabelludo de éste y dejó que le creciera el cabello antes de enviarlo a realizar la entrega. Las técnicas modernas son conceptualmente las mismas, sólo que tienen un mayor ancho de banda y una latencia menor.

Como ejemplo, considere la figura 8-55(a). Esta fotografía, tomada en Kenia, contiene tres cebras que están contemplando un árbol de acacia. La figura 8-55(b) parece ser la misma foto de las tres cebras y el árbol de acacia, pero tiene una atracción extra. Contiene todo el texto de cinco obras de Shakespeare: *Hamlet*, *El Rey Lear*, *Macbeth*, *El Mercader de Venecia* y *Julio César*. Juntas, estas obras tienen un tamaño de 700 KB.

¿Cómo funciona este canal esteganográfico? La imagen a color original es de 1024×768 píxeles. Cada píxel consiste en tres números de 8 bits, cada uno para la intensidad de rojo, verde y azul de ese píxel. El color del píxel se forma por la superposición lineal de los tres colores. El método de codificación esteganográfica utiliza como canal secreto el bit de orden menor de cada valor de color RGB. De esta manera, cada píxel tiene espacio para 3 bits de información secreta,



(a)



(b)

Figura 8-55. (a) Tres cebras y un árbol. (b) Tres cebras, un árbol y todo el texto de cinco obras de William Shakespeare.

uno en el valor de rojo, otro en el valor de verde y otro más en el de azul. En una imagen de este tamaño se pueden almacenar hasta $1024 \times 768 \times 3$ bits o 294,912 bytes de información secreta.

Todo el texto de las cinco obras y una noticia corta suman 734,891 bytes. Este texto primero se comprimió a casi 274 KB mediante un algoritmo de compresión estándar. Después la salida comprimida se encriptó utilizando IDEA y se insertó en los bits de orden menor de cada valor de color. Como puede verse (o más bien, no puede verse), la información es completamente invisible. También es invisible en la versión a todo color de la foto. El ojo no puede distinguir con facilidad los colores de 21 bits de los de 24 bits.

Al ver las dos imágenes en blanco y negro con una resolución baja no se distingue con facilidad el poder de esta técnica. Para tener una mejor idea de cómo funciona la esteganografía, el autor ha preparado una demostración, que incluye la imagen de alta resolución a todo color de la figura 8-55(b) con las cinco obras incrustadas en ella. En el sitio Web del libro se incluye la demostración y las herramientas para insertar y extraer texto en imágenes.

Para utilizar la esteganografía para la comunicación no detectada, los disidentes podrían crear un sitio Web que contenga imágenes políticamente correctas, como fotografías de un gran líder, de deportes locales, de estrellas de películas y de televisión, etcétera. Por supuesto, las imágenes contendrán mensajes esteganográficos. Si los mensajes primero se comprimieran y después se desencriptaran, incluso alguien que sospechara su presencia tendría mucha dificultad para distinguir dichos mensajes del ruido blanco. Por supuesto, las imágenes deben ser digitalizaciones recientes; copiar una imagen de Internet y cambiar algunos de los bits produce una revelación involuntaria.

Las imágenes no son el único medio para los mensajes esteganográficos. Los archivos de audio también funcionan bien. Este tipo de archivos tiene un ancho de banda esteganográfico alto. Incluso el orden de las etiquetas de un archivo HTML puede llevar información.

Aunque hemos examinado la esteganografía en el contexto de la libertad de expresión, tiene varios usos. Uno de los más comunes es para que los dueños de imágenes codifiquen mensajes secretos en ellos que indiquen sus derechos de propiedad. Si una imagen de éstas es robada y se coloca en un sitio Web, el propietario legal puede revelar el mensaje esteganográfico en la corte para probar a quién pertenece esa imagen. Esta técnica se conoce como **watermarking** (**marca de agua**) y se describe en (Piva y cols., 2002).

Para obtener mayor información sobre la esteganografía, vea (Artz, 2001; Johnson y Jajoda, 1998; Katzenbeisser y Petitcolas, 2000, y Wayner, 2002).

8.10.3 Derechos de autor

La privacidad y la censura son sólo dos áreas en las que la tecnología se encuentra con la política pública. Una tercera son los **derechos de autor.** Éstos son el otorgamiento a los creadores de la **IP** (**propiedad intelectual),** incluyendo a los escritores, artistas, compositores, músicos, fotógrafos, cinematógrafos, coreógrafos, entre otros, del derecho exclusivo para explotar su IP por algún tiempo, generalmente durante la vida del autor más 50 o 75 años en el caso de la propiedad corporativa. Después de que expiran los derechos de autor de algún trabajo, pasa a ser del dominio público y cualquiera puede utilizarlo o venderlo como lo desee. Por ejemplo, el Gutenberg Project (*www.promo.net/pg*) ha colocado en Web miles de trabajos de dominio público (de Shakespeare, Twain, Dickens). En 1998, el Congreso de los Estados Unidos extendió por 20 años más los derechos de autor en ese país por solicitud de Hollywood, que afirmó que si no se otorgaba una extensión, nadie crearía nada más. En contraste, las patentes sólo duran 20 años y las personas aún siguen inventando cosas.

Los derechos de autor dieron de qué hablar cuando Napster, un servicio de intercambio de música, tenía 50 millones de miembros. Aunque Napster realmente no copiaba la música, las cortes aseveraron que el hecho de que mantuviera una base de datos de quien tenía las canciones era infracción contributoria, es decir, Napster ayudaba a otras personas a infringir la ley. Si bien nadie alega que los derechos de autor son una mala idea (aunque muchas personas alegan que el término es demasiado tiempo, lo que favorece a las grandes corporaciones y no a las públicas), la siguiente generación de compartición de música ya está provocando problemas mayores de ética.

Por ejemplo, considere una red de igual a igual en la que las personas comparten archivos legales (música del dominio público, vídeos domésticos, pistas religiosas que no son pistas comerciales, etcétera) algunos de los cuales tal vez tengan derechos de autor. Suponga que todos están en línea todo el día mediante ADSL o cable. Cada máquina tiene un índice de lo que hay en el disco duro, más una lista de otros miembros. Alguien que esté buscando un elemento específico puede elegir un miembro al azar y ver si éste tiene tal elemento. Si no lo tiene, puede verificar a todos los miembros que se encuentran en la lista de esa persona y, si no, a todos los miembros que se encuentren en las listas de dichos miembros, y así sucesivamente. Las computadoras son muy buenas para este tipo de trabajo. Cuando encuentra el elemento, el solicitante simplemente lo copia.

Si el trabajo tiene derechos de autor, es probable que el solicitante esté infringiendo la ley (aunque para las transferencias internacionales, la pregunta de la ley de quién se aplica no es clara). Pero, ¿qué sucede con el proveedor? ¿Es un crimen mantener la música por la que se ha pagado y

que se ha descargado legalmente en su disco duro donde otros pueden encontrarla? Si usted tiene una cabina abierta en el país y un ladrón de IP entra con una computadora portátil y un digitalizador, copia un libro que tiene derechos de autor y se va de manera furtiva, ¿*usted* es culpable por no poder proteger los derechos de autor de alguien más?

Pero hay más problemas relacionados con los derechos de autor. En la actualidad hay una gran batalla entre Hollywood y la industria de la computación. El primero quiere protección estricta de toda la propiedad intelectual y la segunda no desea ser el policía de Hollywood. En octubre de 1998, el Congreso aprobó la **DMCA** (**Ley de Propiedad Intelectual para el Milenio Digital**) que considera un crimen evadir cualquier mecanismo de protección presente en un trabajo con derechos de autor o indicar a otros cómo evadirlo. En la Unión Europea se está estableciendo una legislación similar. Si bien nadie piensa que debería permitirse que los piratas del Lejano Oriente duplicaran los trabajos con derechos de autor, muchas personas piensan que la DMCA desplaza por completo el balance entre los intereses de los propietarios con derechos de autor y los públicos.

Por ejemplo, en septiembre de 2000, un consorcio de la industria de la música encargado de construir un sistema irrompible para vender música en línea patrocinó un concurso en el que invitaba a las personas a que trataran de romper el sistema (que es precisamente lo que debería hacerse con cualquier sistema de seguridad). Un equipo de investigadores de seguridad de diversas universidades, dirigidas por el profesor Edward Felten de Princeton, aceptó el desafío y rompió el sistema. Después, este equipo escribió un documento sobre sus descubrimientos y lo emitió a la conferencia de seguridad USENIX, en donde se le sometió a evaluación por expertos y fue aceptado. Antes de que el documento se presentara, Felten recibió una carta de la Recording Industry Association of America, la cual amenazaba con demandarlos apoyándose en la DMCA si publicaban el documento.

Su respuesta fue llevar a juicio pidiendo a la corte federal que reglamentara sobre si el publicar documentos científicos sobre investigación de seguridad era todavía legal. Temiendo que la corte fallara en contra de ellos, la industria retiró su amenaza y la corte desechó la demanda de Felten. Sin duda la industria estaba motivada por la debilidad de su caso: habían invitado a las personas a que trataran de romper su sistema y después trataron de demandar a algunas de ellas por aceptar el desafío. Una vez retirada la amenaza, el documento se publicó (Craver y cols., 2001). Una nueva confrontación es virtualmente cierta.

Un asunto relacionado es la extensión de la **doctrina de uso razonable,** que ha sido establecida por fallos de la corte en varios países. Esta doctrina predica que los compradores de un trabajo con derechos de autor tienen ciertos derechos limitados para copiar el trabajo, incluyendo el derecho de citar partes de él para propósitos científicos, utilizarlo como material de enseñanza en escuelas o colegios y, en algunos casos, realizar copias de respaldo para uso personal en caso de que el medio original falle. Las pruebas de lo que constituye uso razonable incluyen (1) si el uso es comercial, (2) qué porcentaje se copia, y (3) el efecto del copiado en las ventas del trabajo. Puesto que el DMCA y las leyes similares dentro de la Unión Europea prohíben la evasión de los esquemas de protección contra copia, estas leyes también prohíben el uso razonable legal. En efecto, la DMCA elimina los derechos históricos de los usuarios para dar más poder a los vendedores de contenido. Es inevitable un enfrentamiento mayor.

Otro desarrollo en los trabajos que eclipsa incluso a la DMCA en su desplazamiento del equilibrio entre los propietarios de los derechos de autor y los usuarios es la **TCPA** (**Alianza para una Plataforma Informática Confiable**), dirigida por Intel y Microsoft. La idea es que el procesador de la CPU y el sistema operativo supervisen de diversas maneras y con cuidado el comportamiento del usuario (por ejemplo, ejecutando música pirateada) y prohíban el comportamiento no deseable. El sistema incluso permite que los propietarios de contenido manipulen en forma remota las PCs de los usuarios para cambiar las reglas cuando se estime necesario. No es necesario decir que las consecuencias sociales de este esquema son inmensas. Es bueno que la industria finalmente esté poniendo atención a la seguridad, pero es lamentable que esté enteramente dirigido a reforzar las leyes de derechos de autor en lugar de tratar con los virus, *crackers*, intrusos y otros problemas de seguridad por lo que la mayoría de la gente está preocupada.

En resumen, en los siguientes años los legisladores y abogados estarán ocupados equilibrando los intereses económicos de los propietarios de derechos de autor con los intereses públicos. El ciberespacio no es muy diferente de la realidad: constantemente está confrontando a un grupo con otro, lo que resulta en batallas por el poder, litigaciones y (con suerte) algún tipo de resolución, por lo menos hasta que aparezca alguna tecnología inquietante.

8.11 RESUMEN

La criptografía es una herramienta que puede utilizarse para mantener confidencial la información y para asegurar su integridad y autenticidad. Todos los sistemas criptográficos modernos se basan en el principio de Kerckhoff de tener un algoritmo conocido públicamente y una clave secreta. Muchos algoritmos criptográficos utilizan transformaciones complejas que incluyen sustituciones y permutaciones para transformar el texto llano en texto cifrado. Sin embargo, si la criptografía cuántica puede hacerse práctica, el uso de rellenos de una sola vez podría proporcionar criptosistemas realmente irrompibles.

Los algoritmos criptográficos pueden dividirse en algoritmos de clave simétrica y algoritmos de clave pública. Los primeros alteran los bits en una serie de rondas parametrizadas por la clave para convertir al texto llano en texto cifrado. En la actualidad los algoritmos de clave simétrica más populares son el triple DES y Rijndael (AES). Éstos pueden utilizarse en modo de libro de código electrónico, modo de encadenamiento de bloque de cifrado, modo de cifrado de flujo, modo de contador, entre otros.

Los algoritmos de clave pública tienen la propiedad de que se utilizan diferentes claves para la encriptación y desencriptación y de que la clave de desencriptación no puede derivarse de la clave de encriptación. Estas propiedades hacen posible publicar la clave pública. El algoritmo principal de clave pública es RSA, el cual deriva su fuerza del hecho de que es muy difícil factorizar números grandes.

Los documentos legales, comerciales y de otro tipo necesitan firmarse. De manera acorde, se han diseñado varios esquemas para las firmas digitales, las cuales utilizan tanto algoritmos de clave simétrica como de clave pública. Por lo general, a los mensajes que se van a firmar se les aplica un

hash mediante algoritmos como MD5 o SHA-1, y después se firman los *hashes* en lugar de los mensajes originales.

El manejo de claves públicas puede realizarse utilizando certificados, los cuales son documentos que enlazan a un personaje principal con una clave pública. Los certificados son firmados por una autoridad de confianza o por alguien aprobado (recursivamente) por una autoridad confiable. La raíz de la cadena se tiene que obtener por adelantado, pero los navegadores por lo general tienen integrados muchos certificados raíz.

Estas herramientas criptográficas se pueden utilizar para asegurar el tráfico de red seguro. IPsec opera en la capa de red, encriptando flujos de paquetes de *host* a *host*. Los *firewalls* pueden filtrar tráfico entrante y saliente de una organización, frecuentemente con base en el protocolo y puerto utilizado. Las redes privadas virtuales pueden simular una red de línea rentada antigua para proporcionar ciertas propiedades de seguridad deseables. Por último, las redes inalámbricas necesitan buena seguridad y WEP de 802.11 no la proporciona, aunque 802.11i debería mejorar las cosas de manera considerable.

Cuando dos partes establecen una sesión, deben autenticarse entre sí y, si es necesario, establecer una clave de sesión compartida. Existen varios protocolos de autenticación, entre ellos algunos que utilizan un tercero confiable, Diffie-Hellman, Kerberos y la criptografía de clave pública.

La seguridad de correo electrónico puede alcanzarse mediante una combinación de las técnicas que hemos estudiado en este capítulo. Por ejemplo, PGP comprime mensajes, después los encripta utilizando IDEA. Envía la clave IDEA encriptada con la clave pública del receptor. Además, también aplica un *hash* al mensaje y envía el *hash* firmado para verificar la integridad del mensaje.

La seguridad en Web también es un tema importante, iniciando con la asignación de nombres segura. DNSsec proporciona una forma de evitar la falsificación de DNS, al igual que los nombres autocertificables. La mayoría de los sitios Web de comercio electrónico utilizan SSL para establecer sesiones autenticadas seguras entre el cliente y el servidor. Se utilizan varias técnicas para tratar con el código móvil, especialmente las cajas de arena y la firma de código.

Internet provoca muchos problemas en los que la tecnología interactúa de manera considerable con la política pública. Algunas de estas áreas incluyen la privacidad, libertad de expresión y derechos reservados.

PROBLEMAS

1. Rompa el siguiente cifrado monoalfabético. El texto llano, que consiste sólo en letras, es un fragmento bien conocido de un poema de Lewis Carroll.

kfd ktbd fzm eubd kfd pzyiom mztx ku kzyg ur bzha kfthcm
ur mftnm zhx mfudm zhx mdzythc pzq ur ezsszcdm zhx gthcm
zhx pfa kfd mdz tm sutythc fuk zhx pfdkfdi ntcm fzld pthcm

sok pztk z stk kfd uamkdim eitdx sdruid pd fzld uoi efzk
rui mubd ur om zid ouk ur sidzkf zhx zyy ur om zid rzk
hu foiia mztx kfd ezindhkdi kfda kfzhgdx ftb boef rui kfzk

2. Rompa el siguiente cifrado de transposición columnar. El texto llano proviene de un libro de texto de computadoras muy popular, por lo que “computadora” es una palabra probable. El texto llano consiste por completo en letras (sin espacios). Para mayor claridad, el texto cifrado está dividido en bloques de cinco caracteres.

   aauan cvlre rurnn dltme aeepb ytust iceat npmey iicgo gorch srsoc
   nntii imiha oofpa gsivt tpsit lbolr otoex

3. Encuentre un relleno de una sola vez de 77 bits que genere el texto “Donald Duck” a partir del texto cifrado de la figura 8-4.

4. La criptografía cuántica requiere tener un arma de fotones que pueda disparar a solicitud un solo fotón que transporte 1 bit. En este problema, calcule cuántos fotones transporta un bit en un enlace de fibra de 100 Gbps. Suponga que la longitud de un fotón es igual a su longitud de onda, que por propósitos de este problema, es 1 micra. La velocidad de la luz en la fibra es de 20 cm/nseg.

5. Si Trudy captura y regenera fotones cuando está en uso la criptografía cuántica, obtendrá algunos fotones erróneos y causará errores en el relleno de una sola vez de Bob. ¿En promedio qué fracción de los bits de relleno de una sola vez de Bob serán erróneos?

6. Un principio fundamental de criptografía indica que todos los mensajes deben tener redundancia. Pero también sabemos que la redundancia ayuda a que un intruso sepa si una clave adivinada es correcta. Considere dos formas de redundancia. Primero, los n bits iniciales de texto llano contienen un patrón conocido. Segundo, los bits n finales del mensaje contienen un *hash* en el mensaje. Desde un punto de vista de seguridad, ¿estos dos son equivalentes? Analice su respuesta.

7. En la figura 8-6, se alternan las cajas P y S. Aunque este arreglo es estético, ¿es más seguro que primero tener todas las cajas P y después todas las S?

8. Diseñe un ataque a DES con base en el conocimiento de que el texto llano consiste exclusivamente en letras ASCII mayúsculas, más espacio, coma, punto, punto y coma, retorno de carro y avance de línea. No se sabe nada sobre los bits de paridad de texto llano.

9. En el texto calculamos que una máquina para romper cifrado con mil millones de procesadores que pueden analizar una clave en 1 picosegundo podría tardar sólo 10^{10} años para romper la versión de 128 bits de AES. Sin embargo, las máquinas actuales podrían tener 1024 procesadores y tardar 1 mseg en analizar una clave, por lo que necesitamos un factor de mejora en rendimiento de 10^{15} sólo para obtener una máquina de rompimiento AES. Si la ley de Moore (el poder de cómputo se duplica cada 18 meses) sigue vigente, ¿cuántos años se necesitarían tan sólo para construir la máquina?

10. AES soporta una clave de 256 bits. ¿Cuántas claves tiene AES-256? Vea si puede obtener algún número en física, química o astronomía de aproximadamente el mismo tamaño. Utilice Internet para buscar números grandes. Elabore una conclusión a partir de su investigación.

11. Suponga que un mensaje se ha encriptado utilizando DES en modo de encadenamiento de bloque de texto cifrado. Un bit de texto cifrado en el bloque C_i se transforma accidentalmente de 0 a un 1 durante la transmisión. ¿Cuánto texto llano se obtendrá como resultado?

12. Ahora considere nuevamente el encadenamiento de bloque de texto cifrado. En lugar de que un solo bit 0 sea transformado en un bit 1, un bit 0 extra se inserta en el flujo de texto cifrado después del bloque C_i. ¿Cuánto texto llano se distorsionará como resultado?

13. Compare el encadenamiento de bloque cifrado con el modo de retroalimentación de cifrado en términos del número de operaciones de encriptación necesarias para transmitir un archivo grande. ¿Cuál es más eficiente y por cuánto?

14. Utilizando el criptosistema de clave pública RSA, con $a = 1$, $b = 2$, etcétera,
(a) Si $p = 7$ y $q = 11$, liste cinco valores legales para d.
(b) Si $p = 13$, $q = 31$ y $d = 7$, encuentre e.
(c) Utilizando $p = 5$, $q = 11$ y $d = 27$, encuentre e y encripte "abcdefghij".

15. Suponga que un usuario, María, descubre que su clave privada RSA ($d1$, $n1$) es la misma que la clave pública RSA ($e2$, $n2$) de otro usuario, Frances. En otras palabras, $d1 = e2$ y $n1 = n2$. ¿María debe considerar cambiar sus claves pública y privada? Explique su respuesta.

16. Considere el uso del modo de contador, como se muestra en la figura 8-15, pero con $IV = 0$. ¿El uso de 0 amenaza la seguridad del cifrado en general?

17. El protocolo de firma de la figura 8-18 tiene la siguiente debilidad. Si Bob falla, podría perder el contenido de su RAM. ¿Qué problemas causa esto y qué puede hacer él para evitarlos?

18. En la figura 8-20 vimos la forma en que Alice puede enviar a Bob un mensaje firmado. Si Trudy reemplaza *P*, Bob puede detectarlo. Pero, ¿qué sucede si Trudy reemplaza tanto *P* como la firma?

19. Las firmas digitales tienen una debilidad potencial debido a los usuarios flojos. En las transacciones de comercio electrónico podría suscribirse un contrato y el usuario podría solicitar la firma de su *hash* SHA-1. Si el usuario no verifica que el contrato y el *hash* correspondan, firmará de manera inadvertida un contrato diferente. Suponga que la mafia trata de explotar esta debilidad para ganar algo de dinero. Los mafiosos establecen un sitio Web no gratuito (por ejemplo, de pornografía, apuestas, etcétera) y piden a los nuevos clientes un número de tarjeta de crédito. Después envían al cliente un contrato en el que estipulan que éste desea utilizar su servicio y pagar con tarjeta de crédito y le pide que lo firme, a sabiendas de que la mayoría de los clientes simplemente firmarán sin verificar que el contrato y el *hash* correspondan. Muestre la forma en que la mafia puede comprar diamantes de un joyero legítimo de Internet y pueda cargarlos a clientes inocentes.

20. Una clase de matemáticas tiene 20 estudiantes. ¿Cuál es la probabilidad de que por lo menos dos estudiantes tengan la misma fecha de nacimiento? Suponga que nadie nació en año bisiesto, por lo que hay posibles 365 fechas de nacimiento.

21. Después de que Ellen confiesa a Marilyn que la engañó en el asunto de Tom, Marilyn decide evitar este problema dictando el contenido de los mensajes futuros en una máquina de dictado y dárselos a su nueva secretaria para que los teclee. A continuación, Marilyn planea examinar los mensajes en su terminal después de que la secretaria los haya tecleado para asegurarse de que contengan sus palabras exactas. ¿La nueva secretaria aún puede utilizar el ataque de cumpleaños para falsificar un mensaje, y de ser así, cómo? *Sugerencia*: Ella puede hacerlo.

22. Considere el intento fallido de Alice para obtener la clave pública de Bob en la figura 8-23. Suponga que Bob y Alice ya comparten una clave secreta, pero Alice aún quiere la clave pública de Bob. ¿Hay ahora una forma para obtenerla de manera segura? De ser así, ¿cómo?

23. Alice se quiere comunicar con Bob, utilizando la criptografía de clave pública. Ella establece una conexión con alguien que espera sea Bob. Le pide su clave pública y él se la envía en texto llano junto con un certificado X.509 firmado por la raíz CA. Alice ya tiene la clave pública de la raíz CA. ¿Qué pasos debe realizar Alice para verificar que está hablando con Bob? Suponga que a Bob no le importa con quién está hablando (por ejemplo, Bob es algún tipo de servicio público).

24. Suponga que un sistema utiliza PKI con base en una jerarquía con estructura de árbol de CAs. Alice desea comunicarse con Bob y recibe un certificado de Bob firmado por un CA *X* después de establecer el canal de comunicación con Bob. Suponga que Alice nunca ha escuchado sobre *X*. ¿Qué pasos debe realizar Alice para verificar que está hablando con Bob?

25. ¿Es posible utilizar en modo de transporte IPsec con AH si alguna de las máquinas está detrás de una caja NAT? Explique su respuesta.

26. Dé una ventaja de HMACs con respecto del uso de RSA para firmar *hashes* SHA-1.

27. Dé una razón por la cual se configuraría un *firewall* para inspeccionar el tráfico entrante. Dé una razón por la cual se configuraría para inspeccionar tráfico saliente. ¿Cree que las inspecciones sean exitosas?

28. En la figura 8-31 se muestra el formato de paquetes WEP. Suponga que la suma de verificación es de 32 bits, calculada mediante la aplicación de OR exclusivos a todas las palabras de 32 bits en la carga útil. También suponga que los problemas con RC4 se corrigen reemplazándolo con un cifrado de flujo que no tiene debilidad y que los de IV se extienden a 128 bits. ¿Hay alguna forma para que un intruso espíe o interfiera con el tráfico sin ser detectado?

29. Suponga que una organización utiliza VPN para conectar de manera segura sus sitios a Internet. ¿Hay alguna necesidad de que un usuario, Jim, de esta organización utilice la encriptación o cualquier otro mecanismo de seguridad para comunicarse con otro usuario, Mary, de la organización?

30. Cambie ligeramente un mensaje del protocolo de la figura 8-34 para hacerlo resistente al ataque de reflexión. Explique por qué funcionaría su modificación.

31. El intercambio de claves de Diffie-Hellman se utiliza para establecer una clave secreta entre Alice y Bob. Alice envía a Bob (719, 3, 191). Bob responde con (543). El número secreto de Alice, *x*, es 16. ¿Cuál es la clave secreta?

32. Si Alice y Bob no se conocen, no comparten secretos ni tienen certificados, de cualquier manera pueden establecer una clave secreta compartida utilizando el algoritmo de Diffie-Hellman. Explique por qué es muy difícil protegerse contra un ataque de hombre en medio.

33. En el protocolo de la figura 8-39, ¿por qué *A* se envía en texto llano junto con la clave de sesión encriptada?

34. En el protocolo de la figura 8-39, señalamos que iniciar cada mensaje de texto llano con 32 bits 0 es un riesgo de seguridad. Suponga que cada mensaje comienza con un número aleatorio por usuario, efectivamente una segunda clave secreta conocida sólo por su usuario y el KDC. ¿Esto elimina el ataque de texto llano conocido? ¿Por qué?

35. En el protocolo Needham-Schroeder, Alice genera dos desafíos, R_A y R_{A2}. Esto parece excesivo. ¿Uno solo no podría haber sido suficiente?

36. Suponga que una organización utiliza Kerberos para la autenticación. En términos de seguridad y disponibilidad de servicio, ¿cuál es el efecto si AS o TGS se desactivan?

37. En el protocolo de autenticación de clave pública de la figura 8-43, en el mensaje 7, R_B se encripta con K_S. ¿Esta encriptación es necesaria, o podría haber sido adecuado regresarla en texto llano? Explique su respuesta.

38. Las terminales de punto de ventas que utilizan tarjetas con banda magnética y códigos PIN tienen una falla fatal: un comerciante malicioso puede modificar su lector de tarjetas para capturar y almacenar toda la información de la tarjeta, así como el código PIN a fin de realizar posteriormente transacciones adicionales (falsas). La siguiente generación de terminales de punto de ventas utilizará tarjetas con una CPU completa, teclado y una pequeña pantalla en la tarjeta. Diseñe un protocolo para este sistema que los comerciantes maliciosos no puedan romper.

39. Dé *dos* razones por las cuales PGP comprime mensajes.

40. Suponiendo que todos en Internet utilizaron PGP, ¿un mensaje PGP puede enviarse a una dirección arbitraria y decodificarse de manera correcta por todos los interesados? Explique su respuesta.

41. El ataque mostrado en la figura 8-47 omite un paso. Éste no es necesario para que el falsificador trabaje, pero incluyéndolo podría reducir la sospecha potencial después del hecho. ¿Cuál es el paso faltante?

42. Se ha propuesto hacer que la falsificación DNS fracase utilizando la predicción ID haciendo que el servidor coloque un ID aleatorio en lugar de utilizar un contador. Discuta los aspectos de seguridad de este método.

43. El protocolo de transporte de datos SSL involucra dos marcas aleatorias, así como una clave premaestra. ¿Cuál valor, en caso de que haya, tiene el uso de las marcas aleatorias?

44. La imagen de la figura 8-55(b) contiene el texto ASCII de cinco obras de Shakespeare. ¿Sería posible ocultar música entre las cebras en lugar de ocultar texto? De ser así, ¿cómo funcionaría y cuánto podría ocultar en esta imagen? Si no es posible, explique por qué.

45. Alice era usuario de un retransmisor de correo anónimo de tipo 1. Ella podía publicar muchos mensajes en su grupo de noticias favorito, *alt.fanclub.alice*, y todos sabían que tales mensajes provenían de Alice porque todos llevaban el mismo pseudónimo. Suponiendo que el retransmisor de correo funcionaba de manera correcta, Trudy no podía suplantar a Alice. Después de que los retransmisores de correo de tipo 1 desaparecieran, Alice cambió a un retransmisor *cypherpunk* e inició un nuevo subproceso en su grupo de noticias. Diseñe una manera para que Alice evite que Trudy la suplante y publique nuevos mensajes en el grupo de noticias.

46. Busque en Internet un caso interesante que involucre la privacidad y escriba un informe de una página sobre él.

47. Busque en Internet algún caso de una corte que involucre los derechos reservados contra el uso razonable y escriba un informe de una página en el que resuma lo que haya encontrado.

48. Escriba un programa que encripte su entrada al aplicar a ésta un OR exclusivo con un flujo de claves. Busque o escriba lo mejor que pueda un generador de números aleatorios para generar el flujo de claves. El programa debe actuar como un filtro, tomando texto llano como entrada estándar y produciendo texto cifrado como salida estándar (y viceversa). El programa debe tomar un parámetro, la clave que alimenta al generador de números aleatorios.

49. Escriba un procedimiento que calcule el *hash* SHA-1 de un bloque de datos. El procedimiento debe tener dos parámetros: un apuntador al búfer de entrada y otro a un búfer de salida de 20 bytes. Para ver la especificación exacta de SHA-1, busque en Internet FIPS 180-1, que es la especificación completa.

9

LISTA DE LECTURAS Y BIBLIOGRAFÍA

Hemos terminado nuestro estudio de redes de computadoras, pero éste es sólo el comienzo. Por falta de espacio, no tratamos muchos temas interesantes con el detalle que se merecen y omitimos otros. Para beneficio de los lectores que se interesan en continuar sus estudios de redes de computadoras, en este capítulo ofrecemos algunas sugerencias de lecturas adicionales y una bibliografía.

9.1. SUGERENCIAS DE LECTURAS ADICIONALES

Hay una extensa literatura sobre todos los aspectos de redes de computadoras. Tres revistas que con frecuencia publican artículos sobre esta área son *IEEE Transactions on Communications, IEEE Journal on Selected Areas in Communications* y *Computer Communication Review.* Otras muchas revistas también contienen artículos ocasionales sobre el tema.

El IEEE también publica tres revistas —*IEEE Internet Computing, IEEE Network Magazine* e *IEEE Communications Magazine*— que contienen encuestas, tutoriales y casos de estudios sobre conexión de redes. Las dos primeras se enfocan principalmente en la arquitectura, los estándares y el software, y la última tiende a la tecnología de comunicaciones (fibra óptica, satélites, etcétera).

Además, hay varias conferencias anuales o semestrales que provocan la creación de ensayos sobre redes y sistemas distribuidos, en particular, *SIGCOMM Annual Conference, The International Conference on Distributed Computer Systems* y *The Symposium on Operating Systems Principles.*

A continuación presentamos una lista de sugerencias de lectura adicional, relacionadas con los capítulos de este libro. La mayoría son tutoriales o encuestas sobre el tema; otras son capítulos de libros de texto.

9.1.1 Introducción y obras generales

Bi y cols., "Wireless Mobile Communications at the Start of the 21st Century".

Siglo nuevo, tecnología nueva. Suena bien. Después de algunas historias sobre los sistemas inalámbricos, aquí se cubren los temas principales, como los estándares, las aplicaciones, Internet y las tecnologías.

Comer, *The Internet Book.*

Cualquiera que desee una introducción fácil a Internet debe buscarla aquí. Comer describe la historia, el crecimiento, la tecnología, los protocolos y servicios de Internet en términos que los principiantes pueden entender, pero abarca tanto material que el libro también es de interés para lectores más técnicos.

Garber, "Will 3G Really Be the Next Big Wireless Technology?"

Se supone que la tercera generación de teléfonos móviles combinará voz y datos y proporcionará tasas de datos de hasta 2 Mbps. Su lanzamiento ha sido un poco lento. En este artículo fácil de leer se cubren las promesas, los problemas, la tecnología, las políticas y los aspectos económicos del uso de la comunicación inalámbrica de banda ancha.

IEEE Internet Computing, enero-febrero de 2000.

La primera edición de *IEEE Internet Computing* en el nuevo milenio hizo exactamente lo que usted esperaría: preguntar a la gente que ayudó a crear Internet en el milenio anterior hacia dónde piensa que irá en el actual. Los expertos son Paul Baran, Lawrence Roberts, Leonard Kleinrock, Stephen Crocker, Danny Cohen, Bob Metcalfe, Bill Gates, Bill Joy, entre otros. Para mejores resultados, espere 500 años, y *entonces* verifique si sus predicciones fueron acertadas.

Kipnis, "Beating the System: Abuses of the Standars Adoption Process".

Los comités de estándares tratan de ser justos y neutrales en cuanto a proveedores pero, por desgracia, hay compañías que tratan de abusar del sistema. Por ejemplo, con frecuencia sucede que una compañía ayuda al desarrollo de un estándar y, una vez que éste se aprueba, anuncia que dicho estándar se basa en una patente que es de su propiedad, y que sólo dará licencias a compañías

de su agrado, a precios que sólo ella determina. Si desea echar un vistazo al lado oscuro de la estandarización, este artículo es un excelente inicio.

Kwok, "A Vision for Residential Broadband Service".

Si desea saber qué pensaba Microsoft en 1995 acerca de la entrega de vídeo a solicitud, este artículo es para usted. Cinco años más tarde la visión era irremediablemente obsoleta. El valor del artículo está en demostrar que incluso personas muy enteradas y bien intencionadas no pueden ver siquiera cinco años a futuro con certeza alguna. Ésta debe ser una lección para todos.

Kyas y Crawford, *ATM Networks.*

Alguna vez ATM fue aclamado como el protocolo de conexión de redes del futuro, y sigue siendo importante en el sistema de la telefonía. Este libro es una guía actualizada acerca del estado actual de ATM, con información detallada sobre los protocolos ATM y cómo se pueden integrar con las redes basadas en IP.

Naughton, *A Brief History of the Future.*

Sin embargo, ¿quién inventó Internet? Muchas personas han reclamado el crédito. Y con todo derecho, puesto que muchas personas tienen que ver en ello de diversas maneras. Esta historia de Internet narra cómo sucedió todo, de una manera amena y encantadora, rebosante de anécdotas, como la de AT&T que dejaba repetidamente bien sentada su creencia de que las comunicaciones digitales no tenían futuro.

Perkins, "Mobile Networking in the Internet".

Si desea un panorama de redes móviles capa por capa, éste es el lugar donde buscar. Se tratan las capas desde la de transporte hasta la física, así como middleware, seguridad y conexión de redes *ad hoc*.

Tegger y Waks, "End-User Perspectives on Home Networking".

Las redes domésticas no son como las corporativas. Las aplicaciones son diferentes (más intensivas en multimedia), el equipo proviene de un amplio rango de proveedores y los usuarios reciben un breve entrenamiento técnico y no toleran fallas. Para enterarse de más, busque aquí.

Varshney y Vetter, "Emerging Mobile and Wireless Networks".

Otra introducción a la comunicación inalámbrica. Comprende las LANs inalámbricas, los ciclos locales inalámbricos y los satélites, así como algo sobre el software y las aplicaciones.

Wetteroth, *OSI Reference Model for Telecommunications.*

Aunque los protocolos OSI en sí ya no se utilizan, el modelo de siete capas ha llegado a ser muy bien conocido. Además de dar una explicación sobre OSI, este libro aplica el modelo a las redes de telecomunicación (a diferencia de las de computadora), y muestra dónde encajan la telefonía común y otros protocolos de voz en la pila de conexión de redes.

9.1.2 La capa física

Abramson, "Internet Access Using VSATs".

Las estaciones pequeñas en tierra se están volviendo más populares tanto para la telefonía rural como para el acceso corporativo a Internet en países desarrollados. Sin embargo, la naturaleza del tráfico en estos dos casos difiere de manera dramática, por lo que se requieren protocolos diferentes para manejar los dos casos. En este artículo, el inventor del sistema ALOHA expone varios métodos de asignación de canales que se pueden utilizar para sistemas VSAT.

Alkhatib y cols., "Wireless Data Networks: Reaching the Extra Mile".

Si desea una introducción rápida a los términos y tecnologías de la conexión de redes inalámbricas, incluyendo el espectro disperso, este documento tutorial es un buen punto de partida.

Azzam y Ransom, *Broadband Access Technologies*.

Aquí el sistema telefónico, las fibras, ADSL, las redes de cable, los satélites, incluso las líneas de energía, se cubren como tecnologías de acceso de red. Entre otros temas se encuentran las redes domésticas, los servicios, el desempeño de redes y los estándares. El libro concluye con la biografía de las principales compañías en el negocio de redes y telecomunicación, pero con la velocidad de cambios que hay en la industria, este capítulo puede tener una vida de estantería más corta que los capítulos de tecnología.

Bellamy, *Digital Telephony.*

En este libro autorizado se encuentra todo lo que siempre quiso saber sobre el sistema telefónico, entre muchas cosas más. Los capítulos sobre transmisión y multiplexión, conmutación digital, fibras ópticas, telefonía móvil y ADSL son particularmente interesantes.

Berezdivin y cols., "Next-Generation Wireless Communications Concepts and Technologies".

Estas personas van un paso adelante de todas las demás. La parte "next generation (siguiente generación)" se refiere a la cuarta generación de redes inalámbricas. Se espera que estas redes proporcionen servicio IP en todas partes con conectividad a Internet transparente, con alto ancho de banda y una excelente calidad de servicio. Estos objetivos se deben alcanzar mediante el uso inteligente del espectro, administración dinámica de recursos y un servicio adaptable. Todo esto ahora suena visionario, pero aproximadamente en 1995 los teléfonos celulares sonaban bastante visionarios.

Dutta Roy, "An Overview of Cable Modem Technology and Market Perspectives".

La televisión por cable ha dejado de ser un sistema sencillo para convertirse en un sistema complejo de distribución de televisión, Internet y telefonía. Estos cambios han afectado en forma considerable la infraestructura de cable. Vale la pena leer este artículo para tener una explicación sobre las plantas, estándares y marketing de cable, con un énfasis en DOCSIS.

Farserotu y Prasad, "A Survey of Future Broadband Multimedia Satellite Systems, Issues, and Trends".

Varios satélites de comunicaciones están en el cielo o en la mesa de dibujo, como Astrolink, Cyberstar, Spaceway, Skybridge, Teledesic e iSky. Éstos utilizan varias técnicas, como tubo doblado y conmutación satelital. Si desea un panorama de los diferentes sistemas y técnicas de satélite, este documento es un buen punto de partida.

Hu y Li, "Satellite-Based Internet: A Tutorial".

El acceso a Internet vía satélite es distinto del acceso por medio de líneas terrestres. No sólo está el problema del retardo, sino que además el enrutamiento y la conmutación son diferentes. En este documento los autores examinan algunos de los problemas relacionados con el uso de satélites para el acceso a Internet.

Joel, "Telecommunications and the IEEE Communications Society".

Para una historia condensada, pero sorprendentemente amplia, de las telecomunicaciones, empezando con el telégrafo y terminando con el estándar 802.11, este artículo es lo que hay que ver. También cubre radio, teléfonos, conmutación análoga y digital, cables submarinos, transmisión digital, ATM, difusión de televisión, satélites, televisión por cable, comunicaciones ópticas, teléfonos móviles, conmutación de paquetes, ARPANET e Internet.

Metcalfe, "Computer/Network Interface Design: Lessons from Arpanet & Ethernet".

Aunque los ingenieros han estado construyendo interfaces de red durante décadas, con frecuencia nos preguntamos si han aprendido algo de toda esta experiencia. En este artículo, el diseñador de Ethernet describe la manera de construir una interfaz de red, y qué se debe hacer una vez construida. No hace concesiones, indicando lo que hizo mal y lo que hizo bien.

Palais, *Fiber Optic Communication,* 3a. ed.

Los libros sobre la tecnología de la fibra óptica tienden a estar orientados al especialista, pero éste es más accesible que la mayoría; cubre las guías de onda, fuentes de luz, detectores de luz, acopladores, modulación, ruido y muchos otros temas.

Pandya, "Emerging Mobile and Personal Communications Systems".

Para obtener una introducción corta y agradable a los sistemas de comunicación personales portátiles, vea este artículo. Una de las nueve páginas contiene una lista de 70 siglas que se usan en las otras ocho páginas.

Sarikaya, "Packet Mode in Wireless Networks: Overview of Transition to Third Generation".

Toda la idea de las redes celulares de la tercera generación está en los datos de transmisión inalámbrica. Si desea ver un panorama de cómo manejan los datos las redes de la segunda generación y cuál será la evolución a la tercera generación, éste es un buen lugar. Entre los temas que se cubren se encuentran GPRS, IS-95B, WCDMA y CDMA2000.

9.1.3 La capa de enlace de datos

Carlson, *PPP Design, Implementation and Debugging,* 2a. ed.

Si a usted le interesa una información detallada sobre todos los protocolos que integran la suite PPP, incluyendo CCP (compresión) y ECP (encriptación), este libro es una buena referencia. Hay un enfoque particular en ANU PPP-2.3, una implementación popular de PPP.

Gravano, *Introduction to Error Control Codes*.

Los errores se infiltran en casi todas las comunicaciones digitales, y se han desarrollado muchos tipos de código para detectarlos y corregirlos. Este libro explica algunos de los más importantes, desde los sencillos códigos lineales de Hamming hasta los campos y los códigos de Galois más complejos. Aun cuando se intenta utilizar el mínimo de álgebra, contiene una gran cantidad de ésta.

Holzmann, *Design and Validation of Computer Protocols.*

Los lectores interesados en los aspectos más formales de los protocolos de enlace de datos (y similares) deben leer este libro. En él se cubren la especificación, el modelado, la exactitud y las pruebas de tales protocolos.

Peterson y Davie, *Computer Networks: A Systems Approach.*

El capítulo 2 contiene material acerca de muchos problemas de enlace de datos, como tramado, detección de errores, protocolos de parada y espera, protocolos de deslizamiento de ventana y las LAN IEEE 802.

Stallings, *Data and Computer Communications.*

El capítulo 7 trata la capa de enlace de datos y cubre el control de flujo, la detección de errores y los protocolos básicos de enlace de datos, como los protocolos de paro y espera y de regreso n. También se cubren los protocolos de tipo HDLC.

9.1.4 La subcapa de control de acceso al medio

Bhagwat, "Bluetooth: Technology for Short-Range Wireless Apps".

Si desea una introducción directa al sistema Bluetooth, éste es un buen lugar de inicio. Se explican los protocolos y perfiles medulares, radio, *piconets* y enlaces, seguidos de una introducción a los diversos protocolos.

Bisdikian, "An Overview of the Bluetooth Wireless Technology".

Al igual que el documento de Bhagwat (el anterior), éste también es un buen punto de partida para aprender más acerca del sistema Bluetooth. Explica las *piconets*, la pila de protocolos y los perfiles, entre otros temas.

Crow y cols., "IEEE 802.11 Wireless Local Area Networks".

Éste es un buen lugar para empezar una introducción sencilla a la tecnología y a los protocolos del estándar 802.11. Se destaca la subcapa MAC y se cubren tanto el control distribuido como

el control centralizado. El documento concluye con algunos estudios de simulación en diversas circunstancias del desempeño del estándar 802.11.

Eklund y cols., "IEEE Standard 802.16: A Technical Overview of the Wireless MAN Air Interface for Broadband Wireless Access".

El ciclo local inalámbrico estandarizado por el IEEE en 2002, como 802.16, puede revolucionar el servicio telefónico, ofreciendo banda ancha para el hogar. En este panorama, los autores explican los principales problemas de tecnología relacionados con este estándar.

Kapp, "802.11: Leaving the Wire Behind".

Esta breve introducción al estándar 802.11 cubre los fundamentos, los protocolos y los estándares relevantes.

Kleinrock, "On Some Principles of Nomadic Computing and Multi-Access Communications".

El acceso inalámbrico sobre un canal compartido es más complejo que tener estaciones cableadas que comparten un canal. Entre otros problemas se encuentran las topologías dinámicas, el enrutamiento y la administración de energía. En este artículo se cubren estos y otros problemas relacionados con el acceso a canales a través de dispositivos móviles inalámbricos.

Miller y Cummins, *LAN Technologies Explained.*

¿Necesita conocer más acerca de las tecnologías que se pueden emplear en una LAN? Este libro cubre la mayoría de ellas, incluyendo FDDI y token ring, así como la siempre popular Ethernet. Aunque en la actualidad son muy raras las nuevas instalaciones de las dos primeras, muchas redes existentes aún las usan, además de que todavía son comunes las redes de anillo (por ejemplo, SONET).

Perlman, *Interconnections,* 2a. ed.

El libro de Perlman es un buen lugar para buscar un tratamiento entretenido y bien documentado sobre los puentes, enrutadores y enrutamiento en general. El autor diseñó los algoritmos que se utilizan en el puente de árbol de expansión IEEE 802, y evidentemente es una de las autoridades mundiales en varios aspectos de la conectividad.

Webb, "Broadband Fixed Wireless Access".

En este documento se analizan el "porqué" y el "cómo" del ancho de banda fijo inalámbrico. La sección del "porqué" argumenta que las personas no quieren una dirección de correo electrónico doméstica, una dirección de correo electrónica de trabajo, números telefónicos separados para el hogar, trabajo y celular, una cuenta de mensajes instantánea y quizás uno o dos números de fax. Más bien desean un sistema integrado único que funcione en todas partes. En la sección de tecnología se destaca la capa física, incluyendo temas como TDD en comparación con FDD, modulación adaptable en comparación con fija y número de portadoras.

9.1.5 La capa de red

Bhatti y Crowcroft, "QoS Sensitive Flows: Issues in IP Packet Handling".

Una de las maneras de conseguir la mejor calidad de servicio de una red es programar con cuidado las salidas de paquetes de cada enrutador. En este documento se explican en detalle varios algoritmos de programación de paquetes, así como problemas relacionados.

Chakrabarti, "QoS Issues in Ad Hoc Wireless Networks".

El enrutamiento de redes *ad hoc* de computadoras portátiles que están cerca unas de otras es bastante difícil sin tener que preocuparse por la calidad del servicio. No obstante, la gente se preocupa por la calidad del servicio, por lo que hay que prestar atención a este tema. En este artículo se exponen la naturaleza de las redes *ad hoc* y algunos de los problemas relacionados con el enrutamiento y la calidad del servicio.

Comer, *Internetworking with TCP/IP,* Vol. 1, 4a. ed.

Comer ha escrito el libro definitivo sobre el conjunto de protocolos TCP/IP. Los capítulos 4 a 11 tratan el IP y los protocolos relacionados de la capa de red. Los otros capítulos se encargan principalmente de las capas superiores, y también vale la pena leerlos.

Huitema, *Routing in the Internet.*

Si quiere saber todo lo que hay que saber sobre el enrutamiento en Internet, éste es el libro para usted. Se tratan con lujo de detalle tanto los algoritmos pronunciables (por ejemplo, RIP, CIDR y MBONE), como los impronunciables (por ejemplo, OSPF, IGRP, EGP y BGP). También están aquí las características nuevas, como multidifusión, IP móvil y reservación de recursos.

Malhotra, *IP Routing.*

Si desea una guía detallada al enrutamiento IP, este libro contiene mucho material. Entre los protocolos cubiertos están RIP, RIP-2, IGRP, EIGRP, OSPF y BGP-4.

Metz, "Differentiated Services".

Las garantías de calidad de servicio son importantes para muchas aplicaciones multimedia. Los servicios integrados y los diferenciados son dos métodos posibles para alcanzarlas. Los dos se explican aquí, pero se resaltan los servicios diferenciados.

Metz, "IP Routers: New Tool for Gigabit Networking".

La mayoría de las referencias al capítulo 5 son acerca de algoritmos de enrutamiento. Ésta es diferente: se refiere a cómo funcionan en la actualidad los enrutadores. Han pasado por un proceso evolutivo desde ser estaciones de trabajo de propósito general hasta ser máquinas de enrutamiento de propósitos altamente especiales. Si desea saber más, este artículo es un buen lugar para empezar.

Nemeth y cols., *UNIX System Administration Handbook.*

Para un cambio de ritmo, el capítulo 13 de este libro aborda la conexión de redes de una manera más práctica que la mayoría de nuestras demás referencias. En lugar de tratar sólo los conceptos abstractos, aquí se proporcionan muchos consejos sobre qué hacer si usted administra una red real actualmente.

Perkins, "Mobile Networking through Mobile IP".

Conforme los dispositivos de computación portátil se vuelven cada vez más comunes, Mobile IP crece en importancia. Este tutorial da una buena introducción a este y otros temas relacionados.

Perlman, *Interconnections: Bridges and Routers,* 2a. ed.

En los capítulos 12 a 15, Perlman describe muchos de los asuntos implicados en el diseño de algoritmos de enrutamiento unidifusión y multidifusión, tanto en las WANs como en las redes de varias LANs, así como su implementación en varios protocolos. Pero la mejor parte del libro es el capítulo 18, en el que el autor destila sus años de experiencia en protocolos de red en un capítulo divertido e informativo.

Puzmanova, *Routing and Switching: Time of Convergence?*

A finales de 1990, algunos proveedores de equipos de conexión de redes empezaron a llamar conmutador a todo, mientras que muchos administradores de redes grandes decían que estaban cambiando de enrutadores a conmutadores. Como implica el título, este libro predice el futuro de los enrutadores y conmutadores, y cuestiona si en realidad están convergiendo.

Royer y Toh, "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks".

El algoritmo de enrutamiento *ad hoc* AODV que analizamos en el capítulo 5 no es el único conocido. Hay otros más, como DSDV, CGSR, WRP, DSR, TORA, ABR, DRP y SRP, que se explican aquí y se comparan entre sí. Desde luego, si planea inventar un nuevo protocolo de enrutamiento, el paso 1 es idear una sigla de tres o cuatro letras.

Stevens, *TCP/IP Illustrated,* Vol. 1.

Los capítulos 3 a 10 proporcionan un estudio detallado de IP y de los protocolos relacionados (ARP, RARP e ICMP), ilustrado con ejemplos.

Striegel y Manimaram, "A Survey of QoS Multicasting Issues".

La multidifusión y la calidad de servicio son dos temas que cada vez toman más importancia conforme empiezan a despegar servicios como radio y televisión por Internet. En esta encuesta documentada, los autores explican cómo los algoritmos de enrutamiento pueden tener en cuenta estos dos problemas.

Yang y Reddy, “A Taxonomy for Congestion Control Algorithms in Packet Switching Networks”.
Los autores han diseñado una clasificación para los algoritmos de control de congestionamiento. Las categorías principales son ciclo abierto con control de origen, ciclo abierto con control de destino, ciclo cerrado con retroalimentación explícita y ciclo cerrado con realimentación implícita. Los autores usan esta clasificación para describir y clasificar 23 algoritmos existentes.

9.1.6 La capa de transporte

Comer, *Internetworking with TCP/IP,* Vol. 1, 4a. ed.
Como dijimos antes, Comer ha escrito la obra definitiva sobre el conjunto de protocolos TCP/IP. El capítulo 12 se refiere al UDP; el capítulo 13 a TCP.

Hall y Cerf, *Internet Core Protocols: The Definitive Guide.*
Si desea obtener su información directamente del origen, éste es el lugar para aprender acerca de TCP. Después de todo, Cerf lo coinventó. El capítulo 7 es una buena referencia de TCP, pues muestra cómo interpretar la información proporcionada por el análisis de protocolos y las herramientas de administración de redes. Otros capítulos tratan UDP, IGMP, ICMP y ARP.

Kurose y Ross, *Computer Networking: A Top-Down Approach Featuring the Internet.*
El capítulo 3 se refiere a la capa de transporte y contiene una buena cantidad de material sobre UDP y TCP. También presenta los protocolos de parada y espera y de regreso n que analizamos en el capítulo 3.

Mogul, “IP Network Performance”.
A pesar del título de este artículo, en su mayoría trata el desempeño de TCP y las redes en general, más que el de IP en particular; está lleno de lineamientos y reglas empíricas útiles.

Peterson y Davie, *Computer Networks: A Systems Approach.*
El capítulo 5 trata sobre UDP, TCP y algunos protocolos relacionados. Aunque brevemente, también se toca el comportamiento de redes.

Stevens, *TCP/IP Illustrated,* Vol. 1.
Los capítulos 17-24 dan un tratamiento detallado de TCP, ilustrado con ejemplos.

9.1.7 La capa de aplicación

Bergholz, “Extending Your Markup: An XML Tutorial”.
Una introducción directa y breve a XML para principiantes.

Berners-Lee y cols., "The World Wide Web".

Una perspectiva de Web y hacia dónde va, de parte de su inventor y de algunos de los colegas de éste en CERN. El artículo se enfoca en la arquitectura de Web, los URLs, HTPP y HTML, así como en las direcciones futuras, y la compara con los otros sistemas de información distribuidos.

Cardellini y cols*., The State-of-the-Art in Locally Distributed Web-Server Systems".*

A medida que Web se hace más popular, algunos sitios Web necesitan tener grandes granjas de servidores para manejar el tráfico. La parte ardua de construir una granja de servidores es distribuir la carga entre las máquinas. Este documento tutorial explica ampliamente ese tema.

Choudbury y cols., "Copyright Protection for Electronic Publishing on Computer Networks".

Aunque numerosos libros y artículos describen los algoritmos criptográficos, pocos describen la manera en que podrían usarse para evitar que los usuarios distribuyan más documentos de los que tienen permitido desencriptar. Este artículo describe varios mecanismos que pueden ayudar a proteger los derechos de los autores en la era electrónica.

Collins, "Carrier Grade Voice over IP".

Si usted ha leído el trabajo de Varshney y cols., y ahora quiere conocer todos los detalles acerca de la voz a través de IP utilizando H.323, éste es un buen lugar para ver. Aunque el libro es largo y detallado, es tutorial por naturaleza y no requiere conocimientos previos de ingeniería telefónica.

Davison, "A Web Caching Primer".

Conforme Web crece, el uso de caché se hace indispensable para un buen desempeño. Si desea una breve introducción al uso de caché de Web, éste es un buen lugar para ver.

Krishnamurthy y Rexford, *Web Protocols and Practice*.

Sería difícil encontrar un libro más extenso sobre todos los aspectos de Web que éste. Abarca clientes, servidores, *proxies* y uso de caché, como usted podía esperar. Sin embargo, también hay capítulos sobre tráfico y medidas de Web, así como otros sobre investigaciones y mejoras actuales a Web.

Rabinovich y Spatscheck, *Web Caching and Replication.*

Éste contiene un tratamiento extenso del uso de caché y la duplicación de Web. Se cubren en gran detalle *proxies*, cachés, extracción previa, redes de entrega de contenido, selección de servidor y mucho más.

Shahabi y cols., "Yima: A Second-Generation Continuous Media Server".

Los servidores multimedia son sistemas complejos que tienen que administrar programación de CPU, colocación de archivos de disco, sincronización de flujos y más. Con el tiempo, la gente

ha aprendido a diseñarlos mejor. En este documento se presenta un panorama de la arquitectura de uno de los sistemas más recientes.

Tittel y cols., *Mastering XHTML.*

Dos libros en un volumen grande, que cubren el nuevo lenguaje de marca estándar de Web. Primero, hay texto que describe HTML, enfocándose sobre todo en la manera en que se diferencia del HTML regular. Después viene una amplia guía de referencias a etiquetas, códigos y caracteres especiales que se utilizan en XHTML 1.0.

Varshney y cols., "Voice over IP".

¿Cómo funciona la voz a través de IP? ¿Va a reemplazar a la red telefónica conmutada pública? Lea y descúbralo.

9.1.8 Seguridad en redes

Anderson, "Why Cryptosystems Fail".

Según Anderson, la seguridad de los sistemas bancarios es mala, pero no debido a que intrusos astutos violen el DES desde sus PCs. Los verdaderos problemas van desde empleados deshonestos (un empleado bancario que cambia la dirección de correo de un cliente a la suya para interceptar el número de tarjeta bancaria y el número de PIN), a errores de programación (dar a todos los clientes el mismo código PIN). Lo que es especialmente interesante es la altanera respuesta que dan los bancos cuando se les confronta ante un problema evidente: "Nuestros sistemas son perfectos y, por lo tanto, todos los errores deben ser causados por errores del cliente o por fraude".

Anderson, *Security Engineering.*

Algo extenso, este libro es una versión de 600 páginas de "Why Cryptosystems Fail". Es más técnico que *Secrets and Lies,* pero menos que *Network Security* (vea más adelante). Después de una introducción a las técnicas básicas de seguridad, capítulos enteros están dedicados a diversas aplicaciones, como seguridad bancaria, comandos y control nucleares, seguridad en impresión, biometría, seguridad física, guerra electrónica, seguridad en telecomunicaciones, comercio electrónico y protección a derechos de autor. La tercera parte del libro se refiere a políticas, administración y evaluación de sistemas.

Artz, "Digital Steganography".

La esteganografía se remonta a la antigua Grecia, donde se fundía cera en tablas limpias de manera que los mensajes secretos se pudieran aplicar a la madera que estaba debajo antes de aplicar la cera nuevamente. En la actualidad se usan técnicas diferentes, pero el objetivo es el mismo. Aquí se explican varias técnicas modernas para ocultar información en imágenes, audio y otros medios de transporte.

Brands, *Rethinking Public Key Infrastructures and Digital Certificates.*

Más que una introducción amplia a los certificados digitales, éste también es un poderoso trabajo de defensa. El autor cree que los sistemas basados en documentos de verificación de identidad

son obsoletos e inútiles, y argumenta que los certificados digitales se pueden usar para aplicaciones como votaciones electrónicas, administración de derechos digitales e incluso como sustitutos de dinero en efectivo. También advierte que sin PKI ni encriptación, Internet podría llegar a ser una herramienta de supervisión a gran escala.

Kaufman y cols. *Network Security,* 2a. ed.

Para obtener más información técnica sobre algoritmos y protocolos de seguridad en redes, este libro autorizado e ingenioso es lo primero que hay que ver. Algoritmos y protocolos de clave pública y secreta, hashes de mensajes, autenticación, Kerberos, PKI, IPsec, SSL/TLS y seguridad en el correo electrónico, se explican amplia y cuidadosamente, con muchos ejemplos. El capítulo 26 sobre el folklore de la seguridad es una verdadera joya. En seguridad, el demonio está en los detalles. Cualquiera que planee diseñar un sistema de seguridad que realmente se usará, aprenderá en este capítulo mucho del consejo del mundo real.

Pohlmann, *Firewall Systems.*

Los firewalls son la primera línea (y la última) de defensa contra salteadores de redes. Este libro explica cómo funcionan y qué hacen, desde el *firewall* más sencillo con base en software diseñado para proteger una PC, hasta las avanzadas aplicaciones de *firewall* que se sitúan entre una red privada y su conexión a Internet.

Schneier, *Applied Cryptography,* 2a. ed.

Este compendio monumental es la peor pesadilla de NSA: un libro único que describe cada algoritmo criptográfico conocido. Para empeorar las cosas (o para mejorarlas, dependiendo de su punto de vista), el libro contiene la mayoría de los algoritmos como programas ejecutables (en C). Y todavía mas, se proporcionan más de 600 referencias de literatura criptográfica. Este libro no es para principiantes, pero si quiere guardar *realmente* en secreto sus archivos, léalo.

Schneier, *Secrets and Lies.*

Si leyó *Applied Cryptography* de pasta a pasta, lo sabrá todo acerca de algoritmos criptográficos. Si luego lee *Secrets and Lies* de pasta a pasta (lo que se puede hacer en mucho menos tiempo), aprenderá que los algoritmos criptográficos no son toda la historia. La mayoría de las debilidades de seguridad no se debe a fallas en los algoritmos o incluso a claves demasiado cortas, sino a fallas en el entorno de la seguridad. Se presentan innumerables ejemplos sobre amenazas, ataques, defensas, contraataques y mucho más. Para una explicación sencilla y fascinante sobre la seguridad de las computadoras en el más amplio sentido, éste es el libro que hay que leer.

Skoudis, *Counter Hack.*

La mejor manera de detener a un *hacker* es pensar como *hacker*. Este libro muestra cómo ven los *hackers* a una red, y argumenta que la seguridad debe ser una función de todo el diseño de la red, no un pensamiento posterior con base en una tecnología específica. Cubre casi todos los ataques comunes, incluyendo los tipos de “ingeniería social” que se aprovechan de los usuarios que no están familiarizados con las medidas de seguridad de computadora.

9.2 BIBLIOGRAFÍA EN ORDEN ALFABÉTICO

ABRAMSON, N., "Development of the ALOHANET", *IEEE Trans. on Information Theory,* vol. IT-31, págs. 119-123, marzo de 1985.

ABRAMSON, N., "Internet Access Using VSATs", *IEEE Commun. Magazine,* vol. 38, págs. 60-68, julio de 2000.

ADAMS, M., y DULCHINOS, D., "OpenCable", *IEEE Commun. Magazine,* vol. 39, págs. 98-105, junio de 2001.

ALKHATIB, H.S.; BAILEY, C.; GERLA, M., y McRAE, J., "Wireless Data Networks: Reaching the Extra Mile", *Computer,* vol. 30, págs. 59-62, diciembre de 1997.

ANDERSON, R.J., "Free Speech Online and Office", *Computer,* vol. 25, págs. 28-30, junio de 2002.

ANDERSON, R.J., *Security Engineering*, Nueva York: Wiley, 2001.

ANDERSON, R.J., "The Eternity Service", *Proc. First Int'l Conf. on Theory and Appl. Of Cryptology,* CTU Publishing House, 1996.

ANDERSON, R.J., "Why Cryptosystems Fail", *Commun. of the ACM,* vol. 37, págs. 32-40, noviembre de 1994.

ARTZ, D., "Digital Steganography", *IEEE Internet Computing,* vol. 5, págs. 75-80, 2001.

AZZAM, A.A., y RANSOM, N., *Broadband Access Technologies,* Nueva York: McGraw-Hill, 1999.

BAKNE, A., y BADRINATH, B.R., "I-TCP: Indirect TCP for Mobile Hosts", *Proc. 15th Int'l Conf. on Distr. Computer Systems,* IEEE, págs. 136-143, 1995.

BALAKRISHNAN, H.; SESHAN, S., y KATZ, R.H., "Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks", *Proc. ACM Mobile Computing and Networking Conf.,* ACM, págs. 2-11, 1995.

BALLARDIE, T.; FRANCIS, P., y CROWCROFT, J., "Core Based Trees (CBT)", *Proc. SIGCOMM '93 Conf.,* ACM, págs. 85-95, 1993.

BARAKAT, C.; ALTMAN, E., y DABBOUS, W., "On TCP Performance in a Heterogeneous Network: A Survey", *IEEE Commun. Magazine,* vol. 38, págs. 40-46, enero de 2000.

BELLAMY, J., *Digital Telephony*, 3a. ed., Nueva York: Wiley, 2000.

BELLMAN, R.E., *Dynamic Programming,* Princeton, Nueva Jersey: Princeton University Press, 1957.

BELSNES, D., "Flow Control in the Packet Switching Networks", *Communications Networks,* Uxbridge, Inglaterra: Online, págs. 349-361, 1975.

BENNET, C.H., y BRASSARD, G., "Quantum Cryptography: Public Key Distribution and Coin Tossing", *Int'l Conf. on Computer Systems and Signal Processing,* págs. 175-179, 1984.

BEREZDIVIN, R.; BREINIG, R., y TOPP, R., "Next-Generation Wireless Communication Concepts and Technologies", *IEEE Commun. Magazine,* vol. 40, págs. 108-116, marzo de 2002.

BERGHEL, H.L., "Cyber Privacy in the New Millennium", *Computer,* vol. 34, págs. 132-134, enero de 2001.

BERGHOLZ, A., "Extending Your Markup: An XML Tutorial", *IEEE Internet Computing,* vol. 4, págs. 74-79, julio-agosto de 2000.

BERNERS-LEE, T.; CAILLIAU, A.; LOUTONEN, A.; NIELSEN, H.F., y SECRET, A., "The World Wide Web", *Commun. of the ACM,* vol. 37, págs. 76-82, agosto de 1994.

BERTSEKAS, D., y GALLAGER, R., *Data Networks,* 2a. ed., Englewood Cliffs, Nueva Jersey: Prentice Hall, 1992.

BHAGWAT, P., "Bluetooth: Technology for Short-Range Wireless Apps", *IEEE Internet Computing,* vol. 5, págs. 96-103, mayo-junio de 2001.

BHARGHAVAN, V.; DEMERS, A.; SHENKER, S., y ZHANG, L., "MACAW: A Media Access Protocol for Wireless LANs", *Proc. SIGCOMM '94 Conf.,* ACM, págs. 212-225, 1994.

BHATTI, S.N., y CROWCROFT, J., "QoS Sensitive Flows: Issues in IP Packet Handling", *IEEE Internet Computing,* vol. 4, págs. 48-57, julio-agosto de 2000.

BI, Q.; ZYSMAN, G.I., y MENKES, H., "Wireless Mobile Communications at the Start of the 21st Century", *IEEE Commun. Magazine,* vol. 39, págs. 110-116, enero de 2001.

BIHAM, E., y SHAMIR, A., "Differential Cryptanalysis of the Data Encryption Standard", *Proc. 17th Ann. Int'l Cryptology Conf.,* Berlín: Springer-Verlag LNCS 1294, págs. 513-525, 1997.

BIRD, R.; GOPAL, I.; HERZBERG, A.; JANSON, P.A.; KUTTEN, S.: MOLVA, R., y YUNG, M., "Systematic Design of a Family of Attack-Resistant Authentication Protocols", *IEEE J. on Selected Areas in Commun.,* vol. 11, págs. 679-693, junio de 1993.

BIRRELL, A.D., y NELSON, B.J., "Implementing Remote Procedure Calls", *ACM Trans. on Computer Systems,* vol. 2, págs. 39-59, febrero de 1984.

BIRYUKOV, A.; SHAMIR, A., y WAGNER, D., "Real Time Cryptanalysis of A5/1 on a PC", *Proc. Seventh Int'l Workshop on Fast Software Encryption,* Berlín: Springer-Verlag LNCS 1978, p. 1, 2000.

BISDIKIAN, C., "An Overview of the Bluetooth Wireless Technology", *IEEE Commun. Magazine,* vol. 39, págs. 86-94, diciembre de 2001.

BLAZE, M., "Protocol Failure in the Escrowed Encryption Standard", *Proc. Second ACM Conf. on Computer and Commun. Security,* ACM, págs. 59-67, 1994.

BLAZE, M., y BELLOVIN, S., "Tapping on My Network Door", *Commun. of the ACM,* vol. 43, p. 136, octubre de 2000.

BOGINENI, K.; SIVALINGAM, K.M., y DOWD, P.W., "Low-Complexity Multiple Access Protocols for Wavelength-Division Multiplexed Photonic Networks", *IEEE Journal on Selected Areas in Commun.,* vol. 11, págs. 590-604, mayo de 1993.

BOLCSKEI, H.; PAULRAJ, A.J.; HARI, K.V.S., y NABAR, R.U., "Fixed Broadband Wireless Access: State of the Art, Challenges, and Future Directions", *IEEE Commun. Magazine,* vol. 39, págs. 100-108, enero de 2001.

BORISOV, N.; GOLDBERG, I., y WAGNER, D., "Intercepting Mobile Communications: The Insecurity of 802.11", *Seventh Int'l Conf. on Mobile Computing and Networking,* ACM, págs. 180-188, 2001.

BRANDS, S., *Rethinking Public Key Infrastructures and Digital Certificates,* Cambridge, Massachusetts: M.I.T. Press, 2000.

BRAY, J., y STURMAN, C.F., *Bluetooth 1.1: Connect without Cables,* 2a. ed., Upper Saddle River, Nueva Jersey: Prentice Hall, 2002.

BREYER, R., y RILEY, S., *Switched, Fast, and Gigabit Ethernet,* Indianápolis, Indiana: New Riders, 1999.

BROWN, S., *Implementing Virtual Private Networks,* Nueva York: McGraw-Hill, 1999.

BROWN, L.; KWAN, M.; PIEPRZYK, J., y SEBERRY, J., "Improving Resistance to Differential Cryptanalysis and the Redesign of LOKI", *ASIACRYPT '91 Abstracts,* págs. 25-30, 1991.

BURNETT, S., y PAINE, S., *RSA Security's Official Guide to Cryptography,* Berkeley, California: Osborne/McGraw-Hill, 2001.

CAPETANAKIS, J.I., "Tree Algorithms for Packet Broadcast Channels", *IEEE Trans. on Information Theory,* vol. IT-25, págs. 505-515, septiembre de 1979.

CARDELLINI, V.; CASALICCHIO, E.; COLAJANNI, M., y YU, P.S., "*The State-of-the-Art in Locally Distributed Web-Server Systems*", *ACM Computing Surveys,* vol. 34, págs. 263 311, junio de 2002.

CARLSON, J., *PPP Design, Implementation and Debugging,* 2a. ed., Boston: Addison-Wesley, 2001.

CERF, V., y KAHN, R., "A Protocol for Packet Network Interconnection", *IEEE Trans. on Commun.,* vol. COM-22, págs. 637-648, mayo de 1974.

CHAKRABARTI, S., "QoS Issues in Ad Hoc Wireless Networks", *IEEE Commun. Magazine,* vol. 39, págs. 142-148, febrero de 2001.

CHASE, J.S.; GALLATIN, A.J., y YOCUM, K.G., "End System Optimizations for High-Speed TCP", *IEEE Commun. Magazine,* vol. 39, págs. 68-75, abril de 2001.

CHEN, B.; JAMIESON, K.; BALAKRISHNAN, H., y MORRIS, R., "Span: An Energy Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks", *ACM Wireless Networks,* vol. 8, septiembre de 2002.

CHEN, K.-C., "Medium Access Control of Wireless LANs for Mobile Computing", *IEEE Network Magazine,* vol. 8, págs. 50-63, septiembre-octubre de 1994.

CHOUDBURY, A.K.; MAXEMCHUK, N.F.; PAUL, S., y SCHULZRINNE, H.G., "Copyright Protection for Electronic Publishing on Computer Networks", *IEEE Network Magazine,* vol. 9, págs. 12-20, mayo-junio de 1995.

CHU, Y.; RAO, S.G., y ZHANG, H., "A Case for End System Multicast", *Proc. Int'l Conf. on Measurements and Modeling of Computer Syst.,* ACM, págs. 1-12, 2000.

CLARK, D.D., "The Design Philosophy of the DARPA Internet Protocols", *Proc. SIGCOMM '88 Conf.,* ACM, págs. 106-114, 1988.

CLARK, D.D., "Window and Acknowledgement Strategy in TCP", RFC 813, julio de 1982.

CLARK, D.D.; DAVIE, B.S.; FARBER, D.J.; GOPAL, I.S.; KADABA, B.K.; SINCOSKIE, W.D.; SMITH, J.M., y TENNENHOUSE, D.L., "The Aurora Gigabit Testbed", *Computer Networks and ISDN Systems,* vol. 25, págs. 599-621, enero de 1993.

CLARK, D.D.; JACOBSON, V.; ROMKEY, J., y SALWEN, H., "An Analysis of TCP Processing Overhead", *IEEE Commun. Magazine,* vol. 27, págs. 23-29, junio de 1989.

CLARK, D.D.; LAMBERT, M., y ZHANG, L., "NETBLT: A High Throughput Transport Protocol", *Proc. SIGCOMM '87 Conf.,* ACM, págs. 353-359, 1987.

CLARKE, A.C., "Extra-Terrestrial Relays", *Wireless World,* 1945.

CLARKE, I.; MILLER, S.G.; HONG, T.W.; SANDBERG, O., y WILEY, B., "Protecting Free Expression Online with Freenet", *IEEE Internet Computing,* vol. 6, págs. 40-49, enero-febrero de 2002.

COLLINS, D., *Carrier Grade Voice over IP,* Nueva York: McGraw-Hill, 2001.

COLLINS, D., y SMITH, C., *3G Wireless Networks,* Nueva York: McGraw-Hill, 2001.

COMER, D.E., *The Internet Book*, Englewood Cliffs, Nueva Jersey: Prentice Hall, 1995.

COMER, D.E., *Internetworking with TCP/IP,* vol. 1, 4a. ed., Englewood Cliffs, Nueva Jersey: Prentice Hall, 2000.

COSTA, L.H.M.K.; FDIDA, S., y DUARTE, O.C.M.B., "Hop by Hop Multicast Routing Protocol", *Proc. 2001 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Commun.,* ACM, págs. 249-259, 2001.

CRAVER, S.A.; WU, M.; LIU, B.; STUBBLEFIELD, A., SWARTZLANDER, B.; WALLACH, D.W.; DEAN, D., y FELTEN, E.W., "Reading Between the Lines: Lessons from the SDMI Challenge", *Proc. 10th USENIX Security Symp.,* USENIX, 2001.

CRESPO, P.M.; HONIG, M.L., y SALEHI, J.A., "Spread-Time Code-Division Multiple Access", *IEEE Trans. on Commun.,* vol. 43, págs. 2139-2148, junio de 1995.

CROW, B.P.; WIDJAJA, I; KIM, J.G., y SAKAI, P.T., "IEEE 802.11 Wireless Local Area Networks", *IEEE Commun. Magazine,* vol. 35, págs. 116-126, septiembre de 1997.

CROWCROFT, J.; WANG, Z.; SMITH, A., y ADAMS, J., "A Rough Comparison of the IETF and ATM Service Models", *IEEE Network Magazine,* vol. 9, págs. 12-16, noviembre-diciembre de 1995.

DABEK, F.; BRUNSKILL, E.; KAASHOEK, M.F.; KARGER, D.; MORRIS, R.; STOICA, R., y BALAKRISHNAN, H., "Building Peer-to-Peer Systems With Chord, a Distributed Lookup Service", *Proc. 8th Workshop on Hot Topics in Operating Systems,* IEEE, págs. 71-76, 2001a.

DABEK, F.; KAASHOEK, M.F.; KARGER, D.; MORRIS, R., y STOICA, I., "Wide-Area Cooperative Storage with CFS", *Proc. 18th Symp. on Operating Systems Prin.,* ACM, págs. 202-215, 2001b.

DAEMEN, J., y RIJMEN, V., *The Design of Rijndael,* Berlín: Springer-Verlag, 2002.

DANTHINE, A.A.S., "Protocol Representation with Finite-State Models", *IEEE Trans. on Commun.,* vol. COM-28, págs. 632-643, abril de 1980.

DAVIDSON, J., y PETERS, J. *Voice over IP Fundamentals,* Indianápolis, Indiana: Cisco Press, 2000.

DAVIE, B., y REKHTER, Y., *MPLS Technology and Applications,* San Francisco: Morgan Kaufmann, 2000.

DAVIS, P.T., y McGUFFIN, C.R., *Wireless Local Area Networks,* Nueva York: McGraw-Hill, 1995.

DAVISON, B.D., "A Web Caching Primer", *IEEE Internet Computing,* vol. 5, págs. 38-45, julio-agosto de 2001.

DAY, J.D., "The (Un)Revised OSI Reference Model", *Computer Commun. Rev.,* vol. 25, págs. 39-55, octubre de 1995.

DAY, J.D., y ZIMMERMANN, H., "The OSI Reference Model", *Proc. of the IEEE,* vol. 71, págs. 1334-1340, diciembre de 1983.

DE VRIENDT, J.; LAINE, P.; LEROUGE, C., y XU, X., "Mobile Network Evolution: A Revolution on the Move", *IEEE Commun. Magazine,* vol. 40, págs. 104-111, abril de 2002.

DEERING, S.E., "SIP: Simple Internet Protocol", *IEEE Network Magazine,* vol. 7, pp. 16-28, mayo-junio de 1993.

DEMERS, A.; KESHAV, S., y SHENKER, S., "Analysis and Simulation of a Fair Queueing Algorithm", *Internetwork: Research and Experience,* vol. 1, págs. 3-26, septiembre de 1990.

DENNING, D.E., y SACCO, G.M., "Timestamps in Key Distribution Protocols", *Commun. of the ACM,* vol. 24, págs. 533-536, agosto de 1981.

DIFFIE, W., y HELLMAN, M.E., "Exhaustive Cryptanalysis of the NBS Data Encryption Standard", *Computer,* vol. 10, págs. 74-84, junio de 1977.

DIFFIE, W., y HELLMAN, M.E., "New Directions in Cryptography", *IEEE Trans. on Information Theory,* vol. IT-22, págs. 644-654, noviembre de 1976.

DIJKSTRA, E.W., "A Note on Two Problems in Connexion with Graphs", *Numer. Math.,* vol. 1, págs. 269-271, octubre de 1959.

DOBROWSKI, G., y GRISE, D., *ATM and SONET Basics,* Fuquay-Varina, Carolina del Norte: APDG Telecom Books, 2001.

DONALDSON, G., y JONES, D., "Cable Television Broadband Network Architectures", *IEEE Commun. Magazine,* vol. 39, págs. 122-126, junio de 2001.

DORFMAN, R., "Detection of Defective Members of a Large Population", *Annals Math. Statistics,* vol. 14, págs. 436-440, 1943.

DOUFEXI, A.; ARMOUR, S.; BUTLER, M.; NIX, A.; BULL, D.; McGEEHAN, J., y KARLSSON, P., "A Comparison of the HIPERLAN/2 and IEEE 802.11A Wireless LAN Standards", *IEEE Commun. Magazine,* vol. 40, págs. 172-180, mayo de 2002.

DURAND, A., "Deploying IPv6", *IEEE Internet Computing,* vol. 5, págs. 79-81, enero-febrero de 2001.

DUTCHER, B., *The NAT Handbook*, Nueva York: Wiley, 2001.

DUTTA-ROY, A., "An Overview of Cable Modem Technology and Market Perspectives", *IEEE Commun. Magazine,* vol. 39, págs. 81-88, junio de 2001.

EASTTOM, C., *Learn JavaScript,* Ashburton, Reino Unido: Wordware Publishing, 2001.

EL GAMAL, T., "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", *IEEE Trans. on Information Theory,* vol. IT-31, págs. 469-472, julio de 1985.

ELHANANY, I.; KAHANE, M., y SADOT, D., "Packet Scheduling in Next-Generation Multiterabit Networks", *Computer,* vol. 34, págs. 104-106, abril de 2001.

ELMIRGHANI, J.M.H., y MOUFTAH, H.T., "Technologies and Architectures for Scalable Dynamic Dense WDM Networks", *IEEE Commun. Magazine,* vol. 38, págs. 58-66, febrero de 2000.

FARSEROTU, J., y PRASAD, R., "A Survey of Future Broadband Multimedia Satellite Systems, Issues, and Trends", *IEEE Commun. Magazine,* vol. 38, págs. 128-133, junio de 2000.

FIORINI, D.; CHIANI, M.; TRALLI, V., y SALATI, C., "Can we Trust HDLC?", *Computer Commun. Rev.,* vol. 24, págs. 61-80, octubre de 1994.

FLOYD, S., y JACOBSON, V., "Random Early Detection for Congestion Avoidance", *IEEE/ACM Trans. on Networking,* vol. 1, págs. 397-413, agosto de 1993.

FLUHRER, S.; MANTIN, I., y SHAMIR, A., "Weakness in the Key Scheduling Algorithm of RC4", *Proc. Eighth Ann. Workshop on Selected Areas in Cryptography,* 2001.

FORD, L.R., Jr., y FULKERSON, D.R., *Flows in Networks*, Princeton, Nueva Jersey: Princeton, University Press, 1962.

FORD, W., y BAUM, M.S., *Secure Electronic Commerce,* Upper Saddle River, Nueva Jersey: Prentice Hall, 2000.

FORMAN, G.H., y ZAHORJAN, J., "The Challenges of Mobile Computing", *Computer,* vol. 27, págs. 38-47, abril de 1994.

FRANCIS, P., "A Near-Term Architecture for Deploying Pip", *IEEE Network Magazine,* vol. 7, págs. 30-37, mayo-junio de 1993.

FRASER, A.G., "Towards a Universal Data Transport System", en *Advances in Local Area Networks,* Kummerle, K.; Tobagi, F., y Limb, J.O. (Eds.), Nueva York: IEEE Press, 1987.

FRENGLE, N., *I-Mode: A Primer,* Nueva York: Hungry Minds, 2002.

GADECKI, C., y HECKERT, C., *ATM for Dummies,* Nueva York: Hungry Minds, 1997.

GARBER, L., "Will 3G Really Be the Next Big Wireless Technology?", *Computer,* vol. 35, págs. 26-32, enero de 2002.

GARFINKEL, S., con SPAFFORD, G., *Web Security, Privacy, and Commerce,* Sebastopol, California: O'Reilly, 2002.

GEIER, J., *Wireless LANs,* 2a. ed., Indianápolis, Indiana: Sams, 2002.

GEVROS, P.; CROWCROFT, J.; KIRSTEIN, P., y BHATTI, S., "Congestion Control Mechanisms and the Best Effort Service Model", *IEEE Network Magazine,* vol. 15, págs. 16-25, mayo-junio de 2001.

GHANI, N., y DIXIT, S., "TCP/IP Enhancements for Satellite Networks", *IEEE Commun. Magazine,* vol. 37, págs. 64-72, 1999.

GINSBURG, D., *ATM: Solutions for Enterprise Networking,* Boston: Addison-Wesley, 1996.

GOODMAN, D.J., "The Wireless Internet: Promises and Challenges", *Computer,* vol. 33, págs. 36-41, julio de 2000.

GORALSKI, W.J., *Introduction to ATM Networking,* Nueva York: McGraw-Hill, 1995.

GORALSKI, W.J., *Optical Networking and WDM,* Nueva York: McGraw-Hill, 2001.

GORALSKI, W.J., *SONET,* 2a. ed., Nueva York: McGraw-Hill, 2000.

GOSSAIN, H., DE MORAIS CORDEIRO y AGRAWAL, D.P., "Multicast: Wired to Wireless", *IEEE Commun. Mag.,* vol. 40, págs. 116-123, junio de 2002.

GRAVANO, S., *Introduction to Error Control Codes*, Oxford, Reino Unido: Oxford University Press, 2001.

GUO, Y., y CHASKAR, H., "Class-Based Quality of Service over Air Interfaces in 4G Mobile Networks", *IEEE Commun. Magazine,* vol. 40, págs. 132-137, marzo de 2002.

HAARTSEN, J., "The Bluetooth Radio System", *IEEE Personal Commun. Magazine,* vol. 7, págs. 28-36, febrero de 2000.

HAC, A., "Wireless and Cellular Architecture and Services", *IEEE Commun. Magazine,* vol. 33, págs. 98-104, noviembre de 1995.

HAC, A., y GUO, L., "A Scalable Mobile Host Protocol for the Internet", *Int'l J. of Network Mgmt,* vol. 10, págs. 115-134, mayo-junio de 2000.

HALL, E., y CERF, V., *Internet Core Protocols: The Definitive Guide,* Sebastopol, California: O'Reilly, 2000.

HAMMING, R.W., "Error Detecting and Error Correcting Codes", *Bell System Tech. J.,* vol. 29, págs. 147-160, abril de 1950.

HANEGAN, K., *Custom CGI Scripting with Perl,* Nueva York: Wiley, 2001.

HARRIS, A., *JavaScript Programming for the Absolute Beginner,* Premier Press, 2001.

HARTE, L.; KELLOGG, S.; DREHER, R., y SCHAFFNIT, T., *The Comprehensive Guide to Wireless Technology,* Fuquay-Varina, Carolina del Norte: APDG Publishing, 2000.

HARTE, L.; LEVINE, R., y KIKTA, R., *3G Wireless Demystified,* Nueva York: McGraw-Hill, 2002.

HAWLEY, G.T., "Historical Perspectives on the U.S. Telephone System", *IEEE Commun. Magazine,* vol. 29, págs. 24-28, marzo de 1991.

HECHT, J., "Understanding Fiber Optics", Upper Saddle River, Nueva Jersey: Prentice Hall, 2001.

HEEGARD, C.; COFFEY, J.T.; GUMMADI, S.; MURPHY, P.A.; PROVENCIO, R.; ROSSIN, E.J.; SCHRUM, S., y SHOEMAKER, M.B., "High-Performance Wireless Ethernet", *IEEE Commun. Magazine,* vol. 39, págs. 64-73, noviembre de 2001.

HELD, G., *The Complete Modem Reference,* 2a. ed., Nueva York: Wiley, 1994.

HELLMAN, M.E., "A Cryptanalytic Time-Memory Tradeoff", *IEEE Trans. on Information Theory,* vol. IT-26, págs. 401-406, julio de 1980.

HILLS, A., "Large-Scale Wireless LAN Design", *IEEE Commun. Magazine,* vol. 39, págs. 98-104, noviembre de 2001.

HOLZMANN, G.J., *Design and Validation of Computer Protocols,* Englewood Cliffs, Nueva Jersey: Prentice Hall, 1991.

HU, Y., y LI, V.O.K., "Satellite-Based Internet Access", *IEEE Commun. Magazine,* vol. 39, págs. 155-162, marzo de 2001.

HU, Y.-C., y JOHNSON, D.B., "Implicit Source Routes for On-Demand Ad Hoc Network Routing", *Proc. ACM Int'l Symp. on Mobile Ad Hoc Networking & Computing,* ACM, págs. 1-10, 2001.

HUANG, V., y ZHUANG, W., "QoS-Oriented Access Control for 4G Mobile Multimedia CDMA Communications", *IEEE Commun. Magazine,* vol. 40, págs. 118-125, marzo de 2002.

HUBER, J.F.; WEILER, D., y BRAND, H., "UMTS, the Mobile Multimedia Vision for IMT-2000: A Focus on Standardization", *IEEE Commun. Magazine,* vol. 38, págs. 129-136, septiembre de 2000.

HUI, J., "A Broadband Packet Switch for Multi-rate Services", *Proc. Int'l Conf. on Commun.,* IEEE, págs. 782-788, 1987.

HUITEMA, C., *Routing in the Internet*, Englewood Cliffs, Nueva Jersey: Prentice Hall, 1995.

HULL, S., *Content Delivery Networks,* Berkeley, California: Osborne/McGraw-Hill, 2002.

HUMBLET, P.A.; RAMASWAMI, R., y SIVARAJAN, K.N., "An Efficient Communication Protocol for High-Speed Packet-Switched Multichannel Networks", *Proc. SIGCOMM '92 Conf.,* ACM, págs. 2-13, 1992.

HUNTER, D.K., y ANDONOVIC, I., "Approaches to Optical Internet Packet Switching", *IEEE Commun. Magazine,* vol. 38, págs. 116-122, septiembre de 2000.

HUSTON, G., "TCP in a Wireless World", *IEEE Internet Computing,* vol. 5, págs. 82-84, marzo-abril de 2001.

IBE, O.C., *Essentials of ATM Networks and Services,* Boston: Addison-Wesley, 1997.

IRMER, T., "Shaping Future Telecommunications: The Challenge of Global Standardization", *IEEE Commun. Magazine,* vol. 32, págs. 20-28, enero de 1994.

IZZO, P., *Gigabit Networks,* Nueva York: Wiley, 2000.

JACOBSON, V., "Congestion Avoidance and Control", *Proc. SIGCOMM '88 Conf.,* ACM, págs. 314-329, 1988.

JAIN, R., "Congestion Control and Traffic Management in ATM Networks: Recent Advances and a Survey", *Computer Networks and ISDN Systems,* vol. 27, noviembre de 1995.

JAIN, R., "Congestion Control in Computer Networks: Issues and Trends", *IEEE Network Magazine,* vol. 4, págs. 24-30, mayo-junio de 1990.

JAIN, R., *FDDI Handbook—High-Speed Networking Using Fiber and Other Media,* Boston: Addison-Wesley, 1994.

JAKOBSSON, M., y WETZEL, S., "Security Weaknesses in Bluetooth", *Topics in Cryptology: CT-RSA 2001,* Berlín: Springer-Verlag LNCS 2020, págs. 176-191, 2001.

JOEL, A., "Telecommunications and the IEEE Communications Society", *IEEE Commun. Magazine,* edición del 50 aniversario, págs. 6-14 y 162-167, mayo 2002.

JOHANSSON, P.; KAZANTZIDIS, M.; KAPOOR, R., y GERLA, M., "Bluetooth: An Enabler for Personal Area Networking", *IEEE Network Magazine,* vol. 15, págs. 28-37, septiembre-octubre de 2001.

JOHNSON, D.B., "Scalable Support for Transparent Mobile Host Internetworking", *Wireless Networks,* vol. 1, págs. 311-321, octubre de 1995.

JOHNSON, H.W., *Fast Ethernet—Dawn of a New Network,* Englewood Cliffs, Nueva Jersey: Prentice Hall, 1996.

JOHNSON, N.F., y JAJODA, S., "Exploring Steganography: Seeing the Unseen", *Computer,* vol. 31, págs. 26-34, febrero de 1998.

KAHN, D., "Cryptology Goes Public", *IEEE Commun. Magazine,* vol. 18, págs. 19-28, marzo de 1980.

KAHN, D., *The Codebreakers,* 2a. ed., Nueva York: Macmillan, 1995.

KAMOUN, F., y KLEINROCK, L., "Stochastic Performance Evaluation of Hierarchical Routing for Large Networks", *Computer Networks,* vol. 3, págs. 337-353, noviembre de 1979.

KAPP, S., "802.11: Leaving the Wire Behind", *IEEE Internet Computing,* vol. 6, págs. 82-85, enero-febrero de 2002.

KARN, P., "MACA—A New Channel Access Protocol for Packet Radio", *ARRL/CRRL Amateur Radio Ninth Computer Networking Conf.,* págs. 134-140, 1990.

KARTALOPOULOS, S., *Introduction to DWDM Technology: Data in a Rainbow,* Nueva York, Nueva York: IEEE Communications Society, 1999.

KASERA, S.K.; HJALMTYSSON, G.; TOWLSEY, D.F., y KUROSE, J.F., "Scalable Reliable Multicast Using Multiple Multicast Channels", *IEEE/ACM Trans. on Networking,* vol. 8, págs. 294-310, 2000.

KATZ, D., y FORD, P.S., "TUBA: Replacing IP with CLNP", *IEEE Network Magazine,* vol. 7, págs. 38-47, mayo-junio de 1993.

KATZENBEISSER, S., y PETITCOLAS, F.A.P., *Information Hiding Techniques for Steganography and Digital Watermarking,* Londres, Artech House, 2000.

KAUFMAN, C.; PERLMAN, R., y SPECINER, M., *Network Security,* 2a. ed., Englewood Cliffs, Nueva Jersey: Prentice Hall, 2002.

KELLERER, W.; VOGEL, H.-J., y STEINBERG, K.-E., "A Communication Gateway for Infrastructure-Independent 4G Wireless Access", *IEEE Commun. Magazine,* vol. 40, págs. 126-131, marzo de 2002.

KERCKHOFF, A., "La Cryptographie Militaire", *J. des Sciences Militaires,* vol. 9, págs. 5-38, enero de 1883 y págs. 161-191, febrero de 1883.

KIM, J.B.; SUDA, T., y YOSHIMURA, M., "International Standardization of B-ISDN", *Computer Networks and ISDN Systems,* vol. 27, págs. 5-27, octubre de 1994.

KIPNIS, J., "Beating the System: Abuses of the Standards Adoptions Process", *IEEE Commun. Magazine,* vol. 38, págs. 102-105, julio de 2000.

KLEINROCK, L., "On Some Principles of Nomadic Computing and Multi-Access Communications", *IEEE Commun. Magazine,* vol. 38, págs. 46-50, julio de 2000.

KLEINROCK, L., y TOBAGI, F., "Random Access Techniques for Data Transmission over Packet-Switched Radio Channels", *Proc. Nat. Computer Conf.,* págs. 187-201, 1975.

KRISHNAMURTHY, B., y REXFORD, J., *Web Protocols and Practice,* Boston: Addison-Wesley, 2001.

KUMAR, V.; KORPI, M., y SENGODAN, S., *IP Telephony with H.323,* Nueva York: Wiley, 2001.

KUROSE, J.F., y ROSS, K.W., *Computer Networking: A Top-Down Approach Featuring the Internet,* Boston: Addison-Wesley, 2001.

KWOK, T., "A Vision for Residential Broadband Service: ATM to the Home", *IEEE Network Magazine,* vol. 9, págs. 14-28, septiembre-octubre de 1995.

KYAS, O., y CRAWFORD, G., *ATM Networks,* Upper Saddle River, Nueva Jersey: Prentice Hall, 2002.

LAM, C.K.M., y TAN, B.C.Y., "The Internet Is Changing the Music Industry", *Commun. of the ACM,* vol. 44, págs. 62-66, agosto de 2001.

LANSFORD, J.; STEPHENS, A, y NEVO, R., "Wi-Fi (802.11b) and Bluetooth: Enabling Coexistence", *IEEE Network Magazine,* vol. 15, págs. 20-27, septiembre-octubre de 2001.

LASH, D.A., *The Web Wizard's Guide to Perl and CGI,* Boston: Addison-Wesley, 2002.

LAUBACH, M.E.; FARBER, D.J., y DUKES, S.D., *Delivering Internet Connections over Cable,* Nueva York: Wiley, 2001.

LEE, J.S., y MILLER, L.E., *CDMA Systems Engineering Handbook,* Londres: Artech House, 1998.

LEEPER, D.G., "A Long-Term View of Short-Range Wireless", *Computer,* vol. 34, págs. 39-44, junio de 2001.

LEINER, B.M.; COLE, R.; POSTEL, J., y MILLS, D., "The DARPA Internet Protocol Suite", *IEEE Commun. Magazine,* vol. 23, págs. 29-34, marzo de 1985.

LEVINE, D.A., y AKYILDIZ, I.A., "PROTON: A Media Access Control Protocol for Optical Networks with Star Topology", *IEEE/ACM Trans. on Networking,* vol. 3, págs. 158-168, abril de 1995.

LEVY, S., "Crypto Rebels", *Wired,* págs. 54-61, mayo-junio de 1993.

LI, J.; BLAKE, C.; DE COUTO, D.S.J.; LEE, H.I., y MORRIS, R., "Capacity of Ad Hoc Wireless Networks", *Proc. 7th Int'l Conf. on Mobile Computing and Networking,* ACM, págs. 61-69, 2001.

LIN, F.; CHU, P., y LIU, M., "Protocol Verification Using Reachability Analysis: The State Space Explosion Problem and Relief Strategies", *Proc. SIGCOMM '87 Conf.,* ACM, págs. 126-135, 1987.

LIN, Y.-D.; HSU, N.-B., y HWANG, R.-H., "QoS Routing Granularity in MPLS Networks", *IEEE Commun. Magazine,* vol. 40, págs. 58-65, junio de 2002.

LISTANI, M.; ERAMO, V., y SABELLA, R., "Architectural and Technological Issues for Future Optical Internet Networks", *IEEE Commun. Magazine,* vol. 38, págs. 82-92, septiembre de 2000.

LIU, C.L., y LAYLAND, J.W., "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment", *Journal of the ACM,* vol. 20, págs. 46-61, enero de 1973.

LIVINGSTON, D., *Essential XML for Web Professionals,* Upper Saddle River, Nueva Jersey: Prentice Hall, 2002.

LOSHIN, P., *IPv6 Clearly Explained,* San Francisco: Morgan Kaufmann, 1999.

LOUIS, P.J., *Broadband Crash Course,* Nueva York: McGraw-Hill, 2002.

LU, W., *Broadband Wireless Mobile: 3G and Beyond,* Nueva York: Wiley, 2002.

MACEDONIA, M.R., "Distributed File Sharing", *Computer,* vol. 33, págs. 99-101, 2000.

MADRUGA, E.L., y GARCIA-LUNA-ACEVES, J.J., "Scalable Multicasting: the Core Assisted Mesh Protocol", *Mobile Networks and Applications,* vol. 6, págs. 151-165, abril de 2001.

MALHOTRA, R., *IP Routing,* Sebastopol, California: O'Reilly, 2002.

MATSUI, M., "Linear Cryptanalysis Method for DES Cipher", *Advances in Cryptology—Eurocrypt '93 Proceedings,* Berlín: Springer-Verlag LNCS 765, págs. 386-397, 1994.

MAUFER, T.A., *IP Fundamentals,* Upper Saddle River, Nueva Jersey: Prentice Hall, 1999.

MAZIERES, D., y KAASHOEK, M.F., "The Design, Implementation, and Operation of an Email Pseudonym Server", *Proc. Fifth Conf. on Computer and Commun. Security,* ACM, págs. 27-36, 1998.

MAZIERES, D.; KAMINSKY, M.; KAASHOEK, M.F., y WITCHEL, E., "Separating Key Management from File System Security", *Proc. 17th Symp. on Operating Systems Prin.,* ACM, págs. 124-139, diciembre de 1999.

McFEDRIES, P., *Using JavaScript,* Indianápolis, Indiana: Que, 2001.

McKENNEY, P.E., y DOVE, K.F., "Efficient Demultiplexing of Incoming TCP Packets", *Proc. SIGCOMM '92 Conf.,* ACM, págs. 269-279, 1992.

MELTZER, K., y MICHALSKI, B., *Writing CGI Applications with Perl,* Boston: Addison-Wesley, 2001.

MENEZES, A.J., y VANSTONE, S.A., "Elliptic Curve Cryptosystems and Their Implementation", *Journal of Cryptology,* vol. 6, págs. 209-224, 1993.

MERKLE, R.C., "Fast Software Encryption Functions", *Advances in Cryptology—CRYPTO '90 Proceedings,* Berlín: Springer-Verlag LNCS 473, págs. 476-501, 1991.

MERKLE, R.C., y HELLMAN, M., "Hiding and Signatures in Trapdoor Knapsacks", *IEEE Trans. on Information Theory,* vol. IT-24, págs. 525-530, septiembre de 1978.

MERKLE, R.C., y HELLMAN, M., "On the Security of Multiple Encryption", *Commun. of the ACM,* vol. 24, págs. 465-467, julio de 1981.

METCALFE, R.M., "Computer/Network Interface Design: Lessons from Arpanet and Ethernet", *IEEE Journal on Selected Areas in Commun.,* vol. 11, págs. 173-179, febrero de 1993.

METCALFE, R.M., "On Mobile Computing", *Byte,* vol. 20, pág. 110, septiembre de 1995.

METCALFE, R.M., y BOGGS; D.R., "Ethernet: Distributed Packet Switching for Local Computer Networks", *Commun. of the ACM*, vol. 19, págs. 395-404, julio de 1976.

METZ, C., "Differentiated Services", *IEEE Multimedia Magazine,* vol. 7, págs. 84-90, julio-septiembre de 2000.

METZ, C., "Interconnecting ISP Networks", *IEEE Internet Computing,* vol. 5, págs. 74-80, marzo-abril de 2001.

METZ, C., "IP Routers: New Tool for Gigabit Networking", *IEEE Internet Computing,* vol. 2, págs. 14-18, noviembre-diciembre de 1998.

MILLER, B.A., y BISDIKIAN, C., *Bluetooth Revealed,* Upper Saddle River, Nueva Jersey: Prentice Hall, 2001.

MILLER, P., y CUMMINS, M., *LAN Technologies Explained,* Woburn, Massachusetts: Butterworth-Heinemann, 2000.

MINOLI, D., *Video Dialtone Technology,* Nueva York: McGraw-Hill, 1995.

MINOLI, D., y VITELLA, M., *ATM & Cell Relay for Corporate Environments,* Nueva York: McGraw-Hill, 1994.

MISHRA, P.P., y KANAKIA, H., "A Hop by Hop Rate-Based Congestion Control Scheme", *Proc. SIGCOMM '92 Conf.,* ACM, págs. 112-123, 1992.

MISRA, A.; DAS, S.; DUTTA, A.; McAULEY, A., y DAS, S., "IDMP-Based Fast Hand-offs and Paging in IP-Based 4G Mobile Networks", *IEEE Commun. Magazine,* vol. 40, págs. 138-145, marzo de 2002.

MOGUL, J.C., "IP Network Performance", en *Internet System Handbook*, Lynch, D.C. y Rose, M.T. (eds.), Boston: Addison-Wesley, págs. 575-675, 1993.

MOK, A.K., y WARD, S.A., "Distributed Broadcast Channel Access", *Computer Networks*, vol. 3, págs. 327-335, noviembre de 1979.

MOY, J., "Multicast Routing Extensions", *Commun. of the ACM,* vol. 37, págs. 61-66, agosto de 1994.

MULLINS, J., "Making Unbreakable Code", *IEEE Spectrum*, págs. 40-45, mayo de 2002.

NAGLE, J., "Congestion Control in TCP/IP Internetworks", *Computer Commun. Rev.,* vol. 14, págs. 11-17, octubre de 1984.

NAGLE, J., "On Packet Switches with Infinite Storage", *IEEE Trans. on Commun.,* vol. COM-35, págs. 435-438, abril de 1987.

NARAYANASWAMI, C.; KAMIJOH, N.; RAGHUNATH, M.; INOUE, T.; CIPOLLA, T.; SANFORD, J.; SCHLIG, E.; VENTKITESWARAN, S.; GUNIGUNTALA, D.; KULKARNI, V., y YAMAZAKI, K., "IBM's Linux Watch: The Challenge of Miniaturization", *Computer,* vol. 35, págs. 33-41, enero de 2002.

NAUGHTON, J., "A Brief History of the Future", Woodstock, Nueva York: Overlook Press, 2000.

NEEDHAM, R.M., y SCHROEDER, M.D., "Authentication Revisited", *Operating Systems Rev.,* vol. 21, pág. 7, enero de 1987.

NEEDHAM, R.M., y SCHROEDER, M.D., "Using Encryption for Authentication in Large Networks of Computers", *Commun. of the ACM,* vol. 21, págs. 993-999, diciembre de 1978.

NELAKUDITI, S., y ZHANG, Z.-L., "A Localized Adaptive Proportioning Approach to QoS Routing", *IEEE Commun. Magazine,* vol. 40, págs. 66-71, junio de 2002.

NEMETH, E.; SNYDER, G.; SEEBASS, S., y HEIN, T.R., *UNIX System Administration Handbook,* 3a. ed., Englewood Cliffs, Nueva Jersey: Prentice Hall, 2000.

NICHOLS, R.K., y LEKKAS, P.C., *Wireless Security,* Nueva York: McGraw-Hill, 2002.

NIST, "Secure Hash Algorithm", U.S. Government Federal Information Processing Standard 180, 1993.

O'HARA, B., y PETRICK, A., *802.11 Handbook: A Designer's Companion,* Nueva York: IEEE Press, 1999.

OTWAY, D., y REES, O., "Efficient and Timely Mutual Authentication", *Operating Systems Rev.,* págs. 8-10, enero de 1987.

OVADIA, S., *Broadband Cable TV Access Networks: from Technologies to Applications,* Upper Saddle River, Nueva Jersey: Prentice Hall, 2001.

PALAIS, J.C., *Fiber Optic Commun.,* 3a. ed., Englewood Cliffs, Nueva Jersey: Prentice Hall, 1992.

PAN, D., "A Tutorial on MPEG/Audio Compression", *IEEE Multimedia Magazine,* vol. 2, págs. 60-74, verano de 1995.

PANDYA, R., "Emerging Mobile and Personal Communication Systems", *IEEE Commun. Magazine,* vol. 33, págs. 44-52, junio de 1995.

PARAMESWARAN, M.; SUSARLA, A., y WHINSTON, A.B., "P2P Networking: An Information-Sharing Alternative", *Computer,* vol. 34, págs. 31-38, julio de 2001.

PARK, J.S., y SANDHU, R., "Secure Cookies on the Web", *IEEE Internet Computing,* vol. 4, págs. 36-44, julio-agosto de 2000.

PARTRIDGE, C.; HUGHES, J., y STONE, J., "Performance of Checksums and CRCs over Real Data", *Proc. SIGCOMM '95 Conf.,* ACM, págs. 68-76, 1995.

PAXSON, V., "Growth Trends in Wide-Area TCP Connections", *IEEE Network Magazine,* vol. 8, págs. 8-17, julio-agosto de 1994.

PAXSON, V., y FLOYD, S., "Wide-Area Traffic: The Failure of Poisson Modeling", *Proc. SIGCOMM '94 Conf.,* ACM, págs. 257-268, 1995.

PEPELNJAK, I., y GUICHARD, J., *MPLS and VPN Architectures,* Indianápolis, Indiana: Cisco Press, 2001.

PERKINS, C.E., *Mobile IP Design Principles and Practices,* Upper Saddle River, Nueva Jersey: Prentice Hall, 1998a.

PERKINS, C.E., "Mobile Networking in the Internet", *Mobile Networks and Applications,* vol. 3, págs. 319-334, 1998b.

PERKINS, C.E., "Mobile Networking through Mobile IP", *IEEE Internet Computing,* vol. 2, págs. 58-69, enero-febrero de 1998c.

PERKINS, C.E. (ed.), *Ad Hoc Networking,* Boston: Addison-Wesley, 2001.

PERKINS, C.E., *RTP: Audio and Video for the Internet,* Boston: Addison-Wesley, 2002.

PERKINS, C.E., y ROYER, E., "Ad-hoc On-Demand Distance Vector Routing", *Proc. Second Ann. IEEE Workshop on Mobile Computing Systems and Applications,* IEEE, págs. 90-100, 1999.

PERKINS, C.E., y ROYER, E., "The Ad Hoc On-Demand Distance-Vector Protocol", en *Ad Hoc Networking,* editado por C. Perkins, Boston: Addison-Wesley, 2001.

PERLMAN, R., *Network Layer Protocols with Byzantine Robustness,* Ph.D. thesis, M.I.T., 1988.

PERLMAN, R., *Interconnections,* 2a. ed., Boston: Addison-Wesley, 2000.

PERLMAN, R., y KAUFMAN, C., "Key Exchange in IPsec", *IEEE Internet Computing,* vol. 4, págs. 50-56, noviembre-diciembre de 2000.

PETERSON, L.L., y DAVIE, B.S., *Computer Networks: A Systems Approach,* San Francisco: Morgan Kaufmann, 2000.

PETERSON, W.W., y BROWN, D.T., "Cyclic Codes for Error Detection", *Proc. IRE,* vol. 49, págs. 228-235, enero de 1961.

PICKHOLTZ, R.L.; SCHILLING, D.L., y MILSTEIN, L.B., "Theory of Spread Spectrum Communication—A Tutorial", *IEEE Trans. on Commun.,* vol. COM-30, págs. 855-884, mayo de 1982.

PIERRE, G.; KUZ, I.; VAN STEEN, M., y TANENBAUM, A.S., "Differentiated Strategies for Replicating Web Documents", *Computer Commun.,* vol. 24, págs. 232-240, febrero de 2001.

PIERRE, G.; VAN STEEN, M., y TANENBAUM, A.S., "Dynamically Selecting Optimal Distribution Strategies for Web Documents", *IEEE Trans. on Computers,* vol. 51, junio de 2002.

PISCITELLO, D.M., y CHAPIN, A.L., *Open Systems Networking: TCP/IP and OSI,* Boston: Addison-Wesley, 1993.

PITT, D.A., “Bridging—The Double Standard”, *IEEE Network Magazine*, vol. 2, págs. 94-95, enero de 1988.

PIVA, A., BARTOLINI, F., y BARNI, M., “Managing Copyrights in Open Networks, *IEEE Internet Computing,* vol. 6, págs. 18-26, mayo-junio de 2002.

POHLMANN, N., *Firewall Systems,* Bonn, Alemania: MITP-Verlag, 2001.

PUZMANOVA, R., *Routing and Switching: Time of Convergence?,* Londres: Addison-Wesley, 2002.

RABINOVICH, M., y SPATSCHECK, O,. *Web Caching and Replication,* Boston: Addison-Wesley, 2002.

RAJU, J., y GARCIA-LUNA-ACEVES, J.J., “Scenario-based Comparison of Source-Tracing and Dynamic Source Routing Protocols for Ad-Hoc Networks”, *ACM Computer Communications Review,* vol. 31, octubre de 2001.

RAMANATHAN, R., y REDI, J., “A Brief Overview of Ad Hoc Networks: Challenges and Directions”, *IEEE Commun. Magazine,* edición de 50 aniversario, págs. 20-22, mayo de 2002.

RATNASAMY, S.; FRANCIS, P.; HANDLEY, M.; KARP, R., y SHENKER, S., “A Scalable Content-Addressable Network”, *Proc. SIGCOMM ’01 Conf.,* ACM, págs. 1161-1172, 2001.

RIVEST, R.L., “The MD5 Message-Digest Algorithm”, RFC 1320, abril de 1992.

RIVEST, R.L., y SHAMIR, A., “How to Expose an Eavesdropper”, *Commun. of the ACM,* vol. 27, págs. 393-395, abril de 1984.

RIVEST, R.L.; SHAMIR, A., y ADLEMAN, L., “On a Method for Obtaining Digital Signatures and Public Key Cryptosystems”, *Commun. of the ACM,* vol. 21, págs. 120-126, febrero de 1978.

ROBERTS, L.G., “Dynamic Allocation of Satellite Capacity through Packet Reservation”, *Proc. NCC,* AFIPS*,* págs. 711-716, 1973.

ROBERTS, L.G., “Extensions of Packet Communication Technology to a Hand Held Personal Terminal”, *Proc. Spring Joint Computer Conference,* AFIPS, págs. 295-298, 1972.

ROBERTS, L.G., “Multiple Computer Networks and Intercomputer Communication”, *Proc. First Symp. on Operating Systems Prin.,* ACM, 1967.

ROSE, M.T., *The Internet Message,* Englewood Cliffs, Nueva Jersey: Prentice Hall, 1993.

ROSE, M.T., *The Simple Book,* Englewood Cliffs, Nueva Jersey: Prentice Hall, 1994.

ROSE, M.T., y McCLOGHRIE, K., *How to Manage Your Network Using SNMP,* Englewood Cliffs, Nueva Jersey: Prentice Hall, 1995.

ROWSTRON, A., y DRUSCHEL, P., “Storage Management and Caching in PAST, a Large-Scale, Persistent Peer-to-Peer Storage Utility”, *Proc. 18th Symp. on Operating Systems Prin.,* ACM, págs. 188-201, 2001a.

ROWSTRON, A., y DRUSCHEL, P., "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Storage Utility", *Proc. 18th Int'l Conf. on Distributed Systems Platforms,* ACM/IFIP, 2001b.

ROYER, E.M., y TOH, C.-K., "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks", *IEEE Personal Commun. Magazine,* vol. 6, págs. 46-55, abril de 1999.

RUIZ-SANCHEZ, M.A.; BIERSACK, E.W., y DABBOUS, W., "Survey and Taxonomy of IP Address Lookup Algorithms", *IEEE Network Magazine,* vol. 15, págs. 8-23, marzo-abril de 2001.

SAIRAM, K.V.S.S.S.S.; GUNASEKARAN, N., y REDDY, S.R., "Bluetooth in Wireless Communication", *IEEE Commun. Mag.,* vol. 40, págs. 90-96, junio de 2002.

SALTZER, J.H.; REED, D.P., y CLARK, D.D., "End-to-End Arguments in System Design", *ACM Trans. on Computer Systems,* vol. 2, págs. 277-288, noviembre de 1984.

SANDERSON, D.W., y DOUGHERTY, D., *Smileys,* Sebastopol, California: O'Reilly, 1993.

SARI, H.; VANHAVERBEKE, F., y MOENECLAEY, M., "Extending the Capacity of Multiple Access Channels", *IEEE Commun. Magazine,* vol. 38, págs. 74-82, enero de 2000.

SARIKAYA, B., "Packet Mode in Wireless Networks: Overview of Transition to Third Generation", *IEEE Commun. Magazine,* vol. 38, págs. 164-172, septiembre de 2000.

SCHNEIER, B., *Applied Cryptography,* 2a. ed., Nueva York: Wiley, 1996.

SCHNEIER, B., "Description of a New Variable-Length Key, 64-Bit Block Cipher [Blowfish]", *Proc. of the Cambridge Security Workshop,* Berlín: Springer-Verlag LNCS 809, págs. 191-204, 1994.

SCHNEIER, B., *E-Mail Security,* Nueva York: Wiley, 1995.

SCHNEIER, B., *Secrets and Lies,* Nueva York: Wiley, 2000.

SCHNORR, C.P., "Efficient Signature Generation for Smart Cards", *Journal of Cryptology,* vol. 4, págs. 161-174, 1991.

SCHOLTZ, R.A., "The Origins of Spread-Spectrum Communications", *IEEE Trans. on Commun.,* vol. COM-30, págs. 822-854, mayo de 1982.

SCOTT, R., "Wide Open Encryption Design Offers Flexible Implementations", *Cryptologia,* vol. 9, págs. 75-90, enero de 1985.

SEIFERT, R., *Gigabit Ethernet,* Boston: Addison-Wesley, 1998.

SEIFERT, R., *The Switch Book,* Boston: Addison-Wesley, 2000.

SENN, J.A., "The Emergence of M-Commerce", *Computer,* vol. 33, págs. 148-150, diciembre de 2000.

SERJANTOV, A., "Anonymizing Censorship Resistant Systems", *Proc. First Int'l Workshop on Peer-to-Peer Systems,* Berlín: Springer-Verlag LNCS, 2002.

SEVERANCE, C., "IEEE 802.11: Wireless Is Coming Home", *Computer,* vol. 32, págs. 126-127, noviembre de 1999.

SHAHABI, C.; ZIMMERMANN, R.; FU, K., y YAO, S.-Y.D., "YIMA: A Second-Generation Continuous Media Server", *Computer,* vol. 35, págs. 56-64, junio de 2002.

SHANNON, C., "A Mathematical Theory of Communication", *Bell System Tech. J.,* vol. 27, págs. 379-423, julio de 1948; y págs. 623-656, octubre de 1948.

SHEPARD, S., *SONET/SDH Demystified,* Nueva York: McGraw-Hill, 2001.

SHREEDHAR, M., y VARGHESE, G., "Efficient Fair Queueing Using Deficit Round Robin", *Proc. SIGCOMM '95 Conf.,* ACM, págs. 231-243, 1995.

SKOUDIS, E., *Counter Hack,* Upper Saddle River, Nueva Jersey: Prentice Hall, 2002.

SMITH, D.K., y ALEXANDER, R.C., *Fumbling the Future,* Nueva York: William Morrow, 1988.

SMITH, R.W., *Broadband Internet Connections,* Boston: Addison Wesley, 2002.

SNOEREN, A.C., y BALAKRISHNAN, H., "An End-to-End Approach to Host Mobility", *Int'l Conf. on Mobile Computing and Networking,* ACM, págs. 155-166, 2000.

SOBEL, D.L., "Will Carnivore Devour Online Privacy", *Computer,* vol. 34, págs. 87-88, mayo de 2001.

SOLOMON, J.D., *Mobile IP: The Internet Unplugged,* Upper Saddle River, Nueva Jersey: Prentice Hall, 1998.

SPOHN, M., y GARCIA-LUNA-ACEVES, J.J., "Neighborhood Aware Source Routing", *Proc. ACM MobiHoc 2001,* ACM, págs. 2001.

SPURGEON, C.E., *Ethernet: The Definitive Guide,* Sebastopol, California: O'Reilly, 2000.

STALLINGS, W., *Data and Computer Communications,* 6a. ed., Upper Saddle River, Nueva Jersey: Prentice Hall, 2000.

STEINER, J.G.; NEUMAN, B.C., y SCHILLER, J.I., "Kerberos: An Authentication Service for Open Network Systems", *Proc. Winter USENIX Conf.,* USENIX, págs. 191-201, 1988.

STEINMETZ, R., y NAHRSTEDT, K., *Multimedia Fundamentals. Vol. 1: Media Coding and Content Processing,* Upper Saddle River, Nueva Jersey: Prentice Hall, 2002.

STEINMETZ, R., y NAHRSTEDT, K., *Multimedia Fundamentals. Vol. 2: Media Proessing and Communications,* Upper Saddle River, Nueva Jersey: Prentice Hall, 2003a.

STEINMETZ, R., y NAHRSTEDT, K., *Multimedia Fundamentals. Vol. 3: Documents, Security, and Applications,* Upper Saddle River, Nueva Jersey: Prentice Hall, 2003b.

STEVENS, W.R., *TCP/IP Illustrated,* Vol. 1, Boston: Addison-Wesley, 1994.

STEVENS, W.R., *UNIX Network Programming, Volume 1: Networking APIs - Sockets and XTI,* Upper Saddle River, Nueva Jersey: Prentice Hall, 1997.

STEWART, R., y METZ, C., "SCTP: New Transport Protocol for TCP/IP", *IEEE Internet Computing,* vol. 5, págs. 64-69, noviembre-diciembre de 2001.

STINSON, D.R., *Cryptography Theory and Practice,* 2a. ed., Boca Ratón, Florida: CRC Press, 2002.

STOICA, I.; MORRIS, R.; KARGER, D.; KAASHOEK, M.F., y BALAKRISHNAN, H., "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications", *Proc. SIGCOMM '01 Conf.,* ACM, págs. 149-160, 2001.

STRIEGEL, A., y MANIMARAN, G., "A Survey of QoS Multicasting Issues", *IEEE Commun. Mag.,* vol. 40, págs. 82-87, junio de 2002.

STUBBLEFIELD, A.; IOANNIDIS, J., y RUBIN, A.D., "Using the Fluhrer, Mantin, and Shamir Attack to Break WEP", *Proc Network and Distributed Systems Security Symp.,* ISOC, págs. 1-11, 2002.

SUMMERS, C.K., *ADSL: Standards, Implementation, and Architecture,* Boca Ratón, Florida: CRC Press, 1999.

SUNSHINE, C.A., y DALAL, Y.K., "Connection Management in Transport Protocols", *Computer Networks,* vol. 2, págs. 454-473, 1978.

TANENBAUM, A.S., *Modern Operating Systems,* Upper Saddle River, Nueva Jersey: Prentice Hall, 2001.

TANENBAUM, A.S., y VAN STEEN, M., *Distributed Systems: Principles and Paradigms,* Upper Saddle River, Nueva Jersey: Prentice Hall, 2002.

TEGER, S., y WAKS, D.J., "End-User Perspectives on Home Networking", *IEEE Commun. Magazine,* vol. 40, págs. 114-119, abril de 2002.

THYAGARAJAN, A.S., y DEERING, S.E., "Hierarchical Distance-Vector Multicast Routing for the MBone", *Proc. SIGCOMM '95 Conf.,* ACM, págs. 60-66, 1995.

TITTEL, E.; VALENTINE, C.; BURMEISTER, M., y DYKES, L., *Mastering XHTML,* Alameda, California: Sybex, 2001.

TOKORO, M., y TAMARU, K., "Acknowledging Ethernet", *Compcon,* IEEE, págs. 320-325, otoño de 1977.

TOMLINSON, R.S., "Selecting Sequence Numbers", *Proc. SIGCOMM/SIGOPS Interprocess Commun. Workshop,* ACM, págs. 11-23, 1975.

TSENG, Y.-C.; WU, S.-L.; LIAO, W.-H., y CHAO, C.-M., "Location Awareness in Ad Hoc Wireless Mobile Networks", *Computer,* vol. 34, págs. 46-51, 2001.

TUCHMAN, W., "Hellman Presents No Shortcut Solutions to DES", *IEEE Spectrum,* vol. 16, págs. 40-41, julio de 1979.

TURNER, J.S., "New Directions in Communications (or Which Way to the Information Age)", *IEEE Commun. Magazine,* vol. 24, págs. 8-15, octubre de 1986.

VACCA, J.R., *I-Mode Crash Course,* Nueva York: McGraw-Hill, 2002.

VALADE, J., *PHP & MySQL for Dummies,* Nueva York: Hungry Minds, 2002.

VARGHESE, G, y LAUCK, T., "Hashed and Hierarchical Timing Wheels: Data Structures for the Efficient Implementation of a Timer Facility", *Proc. 11th Symp. on Operating Systems Prin.,* ACM, págs. 25-38, 1987.

VARSHNEY, U.; SNOW, A.; McGIVERN, M., y HOWARD, C., *"Voice over IP", Commun. of the ACM,* vol. 45, págs. 89-96, 2002.

VARSHNEY, U., y VETTER, R., "Emerging Mobile and Wireless Networks", *Commun. of the ACM,* vol. 43, págs. 73-81, junio de 2000.

VETTER, P.; GODERIS, D.; VERPOOTEN, L., y GRANGER, A., "Systems Aspects of APON/VDSL Deployment", *IEEE Commun. Magazine,* vol. 38, págs. 66-72, mayo de 2000.

WADDINGTON, D.G., y CHANG, F., "Realizing the Transition to IPv6", *IEEE Commun. Mag.,* vol. 40, págs. 138-148, junio de 2002.

WALDMAN, M.; RUBIN, A.D., y CRANOR, L.F., "Publius: A Robust, Tamper-Evident, Censorship-Resistant, Web Publishing System", *Proc. Ninth USENIX Security Symp.,* USENIX, págs. 59-72, 2000.

WANG, Y., y CHEN, W., "Supporting IP Multicast for Mobile Hosts", *Mobile Networks and Applications,* vol. 6, págs. 57-66, enero-febrero de 2001.

WANG, Z., *Internet QoS,* San Francisco: Morgan Kaufmann, 2001.

WARNEKE, B.; LAST, M.; LIEBOWITZ, B., y PISTER, K.S.J.: "Smart Dust: Communicating with a Cubic Millimeter Computer", *Computer,* vol. 34, págs. 44-51, enero de 2001.

WAYNER, P., *Disappearing Cryptography: Information Hiding, Steganography, and Watermarking,* 2a. ed., San Francisco: Morgan Kaufmann, 2002.

WEBB, W., "Broadband Fixed Wireless Access as a Key Component of the Future Integrated Communications Environment", *IEEE Commun. Magazine,* vol. 39, págs. 115-121, septiembre de 2001.

WEISER, M., "Whatever Happened to the Next Generation Internet?", *Commun. of the ACM,* vol. 44, págs. 61-68, septiembre de 2001.

WELTMAN, R., y DAHBURA, T., *LDAP Programming with Java,* Boston: Addison-Wesley, 2000.

WESSELS, D., *Web Caching,* Sebastopol, California: O'Reilly, 2001.

WETTEROTH, D., *OSI Reference Model for Telecommunications,* Nueva York: McGraw-Hill, 2001.

WILJAKKA, J., "Transition to IPv6 in GPRS and WCDMA Mobile Networks", *IEEE Commun. Magazine,* vol. 40, págs. 134-140, abril de 2002.

WILLIAMSON, H., *XML: The Complete Reference,* Nueva York: McGraw-Hill, 2001.

WILLINGER, W.; TAQQU, M.S.; SHERMAN, R., y WILSON, D.V., "Self-Similarity through High Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level", *Proc. SIGCOMM '95 Conf.,* ACM, págs. 100-113, 1995.

WRIGHT, D.J., *Voice over Packet Networks,* Nueva York: Wiley, 2001.

WYLIE, J.; BIGRIGG, M.W.; STRUNK, J.D.; GANGER, G.R.; KILICCOTE, H., y KHOSLA, P.K., "Survivable Information Storage Systems", *Computer,* vol. 33, págs. 61-68, agosto de 2000.

XYLOMENOS, G.; POLYZOS, G.C.; MAHONEN, P., y SAARANEN, M., "TCP Performance Issues over Wireless Links", *IEEE Commun. Magazine,* vol. 39, págs. 52-58, abril de 2001.

YANG, C.-Q., y REDDY, A.V.S., "A Taxonomy for Congestion Control Algorithms in Packet Switching Networks", *IEEE Network Magazine,* vol. 9, págs. 34-45, julio-agosto de 1995.

YUVAL, G., "How to Swindle Rabin", *Cryptologia,* vol. 3, págs. 187-190, julio de 1979.

ZACKS, M., "Antiterrorist Legislation Expands Electronic Snooping", *IEEE Internet Computing,* vol. 5, págs. 8-9, noviembre-diciembre de 2001.

ZADEH, A.N.; JABBARI, B.; PICKHOLTZ, R., y VOJCIC, B., "Self-Organizing Packet Radio Ad Hoc Networks with Overlay (SOPRANO)", *IEEE Commun. Mag.,* vol. 40, págs. 149-157, junio de 2002.

ZHANG, L., "Comparison of Two Bridge Routing Approaches", *IEEE Network Magazine,* vol. 2, págs. 44-48, enero-febrero de 1988.

ZHANG, L., "RSVP: A New Resource ReSerVation Protocol", *IEEE Network Magazine,* vol. 7, págs. 8-18, septiembre-octubre de 1993.

ZHANG, Y., y RYU, B., "Mobile and Multicast IP Services in PACS: System Architecture, Prototype, and Performance", *Mobile Networks and Applications,* vol. 6, págs. 81-94, enero-febrero de 2001.

ZIMMERMANN, P.R., *The Official PGP User's Guide,* Cambridge, Massachusetts: M.I.T. Press, 1995a.

ZIMMERMANN, P.R., *PGP: Source Code and Internals,* Cambridge, Massachusetts: M.I.T. Press, 1995b.

ZIPF, G.K., *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology,* Boston: Addison-Wesley, 1949.

ZIV, J., y LEMPEL, Z., "A Universal Algorithm for Sequential Data Compression", *IEEE Trans. on Information Theory,* vol. IT-23, págs. 337-343, mayo de 1977.

ÍNDICE

Números

2.5G, sistema de telefonía móvil, 168
3G (*vea* tercera generación, sistema telefónico móvil de)
4B/5B, 285
8B/6T, 285
8B/10B, 289
10Base-*x*, 272
100Base-*x*, 285
802 (*vea* IEEE 802.*x*)
802.3 (*vea* Ethernet)
802.15 (*vea* Bluetooth)
802.16 (*vea* IEEE 802.16)
1000Base-*x*, 288

A

A, B, C y D, clases, 437
AAL (*vea* Capa de Adaptación de ATM)
AAL-SAP, 494
Abramson, Norman, 65-66
Acceso aleatorio, canal de, 162, 247
Acceso Múltiple con Detección de Portadora
  con detección de colisiones, 255-258
  con evitación de colisiones, 296-297
  no persistente, 256
  persistente-1, 255
  persistente-p, 256
Acceso Múltiple con Prevención de Colisiones, 269-270
Acceso Múltiple de División de Código, 162-166
Acceso múltiple, protocolos de, 251-270
ACL (*vea* Asíncrono no Orientado a la Conexión, enlace)
Active Server Pages, 646
ActiveX, control, 650, 817-818
Actores principales, 731
Ad hoc, red, 68, 375-380
ADC (*vea* Convertidor Analógico a Digital)
ADCCP (*vea* Procedimiento Avanzado de Control de Comunicación de Datos)
Administración de conexiones, modelado de, 541-543
  máquina de estados, 486
Administración de temporizadores, TCP, 550-553
Admisión, control de, 389, 406-408
ADSL (*vea* Línea Digital Asimétrica de Suscriptor)
AES (*vea* Estándar de Encriptación Avanzada)
Agencia de Proyectos de Investigación Avanzada, 51
Agencia Nacional de Seguridad, 740
Agente(s)
  de base, 373
  de transferencia, correo electrónico, 590
  de usuario, correo electrónico, 590
  foráneos, 373
Aglomeración instantánea, 660
AH (*vea* encabezado, de autenticación)
Alambre de cobre, comparación con la fibra óptica, 98-99

Algoritmo Internacional de Encriptación de Datos, 799
Algoritmo(s)
adaptativos, 351
basados en flujo, 409-412
de codificación, vídeo, 696
de decodificación, vídeo, 696
de enrutamiento adaptativos, 424
seguro
de *Hash*, 761-762
de SHA-1, 761-762
Alianza para una Plataforma Informática Confiable, 828
Alice, 731
Alineación, cable de televisión, 174
Almacenamiento en caché jerárquico, 658-659
Almacenamiento y reenvío, conmutación de paquetes, 20, 344
ALOHA, 251-255
puro, 251-254
ranurado, 254-255
Amplitud, modulación de, 126
AMPS (*vea* Sistema Avanzado de Telefonía Móvil)
Ancho de banda, 88
Ancho de banda-retardo, producto, 559
Anclas de confianza, 770
Anderson, Ross, 742
Andreessen, Mark, 57, 611
ANS (*vea* Redes y Servicios Avanzados)
ANSI (*vea* Instituto Nacional de Estándares Nacionales)
ANSNET, 55
AODV (*vea* Vector de Distancia *ad hoc* bajo Demanda)
Aplicación auxiliar, 617
Applets, 650
Aprendizaje hacia atrás, 323
Árbol(es)
de expansión, 368
puentes con, 323-325
de núcleo, 372
sumidero, 352
Archivos, transferencia de, 57
Área(s), 456
de acceso y transporte local, 122
enrutadores, 457
red de, personal, 15
Armónicos, 86
ARP
gratuito, 463
proxy, 452
vea también Protocolo de Resolución de Direcciones
ARPA (*vea* Agencia de Proyectos de Investigación Avanzada)
ARPANET, 50-54
algoritmos de enrutamiento, 357, 454
ARQ (*vea* Solicitud Automática de Repetición)
Arquitectura de red, 28-30
Arranque lento
algoritmo de Jacobson, 549-550
TCP, 549-550
Arreglo redundante de discos baratos, 708
Arrendamiento, 454
AS (*vea* sistema autónomo)
ASCII, armadura, 598
Asequibilidad, análisis de, 230
Asignación del canal, problema de, 248-251, 337
Asíncrono no Orientado a la Conexión, enlace, 315
ASN.1 (*vea* Notación de Sintaxis Abstracta 1)
Asociación para Seguridad en Internet y Administración de Claves, 773
Asociación, servicios, 301
ASP (*vea* Active Server Pages)
Ataque, seguridad
consumo de energía, 751
DDoS, 778
de cumpleaños, 763-765, 782
de hombre en medio, 792
de la brigada de bomberos, 792
de reflexión, 787-790
de repetición, 794
DoS, 778
reutilización de flujo de claves, 749
sólo texto cifrado, 727
temporización, 751
texto llano conocido, 727
texto llano seleccionado, 727
Atenuación, 125
ATM
redes de, 61, 65, 417-418
subcapa dependiente del medio físico, 64
vea también Modo de Transferencia Asíncrona
Atributo
certificado, 767
HTML, 630
Audio, 674-692
compresión de, 676-679
codificación porcentual, 677
enmascaramiento de frecuencia, 677
enmascaramiento temporal, 677
MP3, 676-679
psicoacústica, 677
introducción al, digital, 674-676
Autenticación, 785
centro de distribución de claves, 793-796
clave
compartida, 786-790
pública, 798-799
Diffie-Hellman, 791-792
HMAC, 790
Kerberos, 796-798
Needham-Schroeder, 794-795
Otway-Rees, 795-796
servicios, 302

Authenticode, 818
Autoridad de Certificación, 766
Autoridades Regionales, 769
Bush, Vannevar, 612
Búsquedas invertidas, 584
Buzones de correo, correo electrónico, 591

B

Banda ancha, 130
Banda ancha inalámbrica (*vea* IEEE 802.16)
Bandas industriales, médicas y científicas, 106, 292-293, 315
Bandera, 189
Baran, Paul, 50
Barker, secuencia, 294
Base
  diagonal, 732
  rectilínea, 732
Base64, codificación, 598
Baudios, 127
Beacon, trama de, 298
Bell, Alexander Graham, 119
Bellman-Ford, algoritmo de enrutamiento, 357-360, 454
Berkeley, *sockets* de, 487-488
BGP (*vea* Protocolo de Puerta de Enlace de Frontera)
Big endian, máquina, 433
Biham, Eli, 742
Bits
  relleno de, 190
  secuencia ortogonal de, 164
Blaatand, Harald, 310
Blanqueamiento, 740
Bloque(s)
  de cifrado, modo de encadenamiento de, 746-747
  irreversible, red de Petri, 232
Blowfish, 751
Bluetooth, 21, 310-317
  aplicaciones, 312-313
  arquitectura, 311
  Asíncrono no Orientado a la Conexión, enlace, 315
  capa de banda base, 315-316
  enlace, 315
  historia, 310
  perfiles, 312-313
  piconet, 311
  pila de protocolos, 313-314
  scatternet, 311
  seguridad, 783-784
  Síncrono Orientado a la Conexión, enlace, 316
Bob, 731
BOC (*vea* Compañías Operativas de Bell)
BOOTP, protocolo, 453
Borrador de Estándar Internacional, 75
Brigada de bomberos, ataque de la, 792
Búfer, almacenamiento en, 506-510

C

CA (*vea* Autoridad de Certificación)
Cable
  coaxial, 92
  módem, 173-175
Cableado, gabinete de, 91
Caché, 657-659
  encabezado
    If-Modified-Since, 659
    Last-Modified, 658-659
  envenenado, 807
  jerárquico, 658-659
  obsoleto, 658-659
Caídas, recuperación de, 511-513
Caja
  de arena, 817
  P, 737
  S, 737
Calidad
  del servicio, 32
    algoritmo de cubeta con goteo, 400-403
    algoritmo de cubeta con *tokens*, 402-405
    algoritmos basados en flujo, 409-412
    almacenamiento en búfer, 399
    basada en clase, 412-415
    calendarización de paquetes, 408-409
    capa de red, 397-417
    conmutación de etiquetas, 415-417
    control de admisión, 406-408
    diferenciada, 412-415
    encolamiento justo, 408-409
    enrutamiento proporcional, 408
    especificación de flujo, 407
    integrada, 409-412
    modelado de tráfico, 399-400
    MPLS, 415-417
    reenvío asegurado, 414-415
    reenvío expedito, 413-414
    requerimientos, 397-398
    reservación de recursos, 405-406
    RSVP, 410-412
    técnicas para alcanzar buena, 398-409
  línea con, de voz, 88
Campo, vídeo, 693
Canal(es)
  asignación
    dinámica de, 249-251
    estática de, 248-249

de fibra, 283
de otorgamiento de acceso, 162
dedicado de control, 161
señalización
por, asociado, 141
por, común, 141
Canalización, 217
Capa, 26
aspectos del diseño, 30-31
de aplicación, 41, 43, 579-720
correo electrónico, 588-611
DNS, 579-588
multimedia, 674-714
World Wide Web, 611-673
de enlace de datos, 38, 183-246
conmutación en la, 317-336
de presentación, 41
de red, 39, 343-480
algoritmos de enrutamiento, 350-384
aspectos de diseño, 343-350
calidad del servicio, 397-417
control de congestión, 384-396
interconectividad, 418-431
Internet, 431-473
de sesión, 40
de transporte, 481-578
acuerdo de tres vías, 500-502
almacenamiento en búfer, 506-510
aspectos del desempeño, 557-573
control de flujo, 506-510
direccionamiento, 493-496
establecimiento de una conexión, 496-502
Internet, 524-556
liberación de una conexión, 502-506
multiplexión, 510-511
protocolo de ejemplo, 513-524
recuperación de caídas, 511-513
seguridad de, 816
servicio, 481-492
TCP, 532-566
UDP, 524-532
del portador, WAP, 664
física, 38, 85-182
IEEE 802.11, 293-295
IEEE 802.16, 306-307
Internet a través de cable, 170-176
medios alámbricos, 90-99
satélites de comunicación, 109-118
sistema telefónico, 118-151
sistema telefónico móvil, 152-169
transmisión inalámbrica, 100-108
Capa de Adaptación de ATM, 64
Capa Inalámbrica de Seguridad de Transporte, 664
Caracteres, relleno de, 189-190
Caritas, 588-589
Carnivore, 13
Casi vídeo bajo demanda, 704
Categoría
3, cableado, 91
5, cableado, 92
CCITT, 72
CD (*vea* comité, borrador de)
CD, audio, 676
CDMA (*vea* Acceso Múltiple de División de Código)
CDMA2000, 167
CdmaOne, 162
CDN (*vea* Redes de Entrega de Contenido)
Celda
HTML, 633
transporte de, 155
duro, 155
suave, 155
Centro de Conmutación Móvil, 155
Centro de Distribución de Claves, 785
Certificación, ruta de, 770
Certificados, 765-771
revocación de, 771
César, cifrado de, 727
CGI (*vea* Interfaz de Puerta de Enlace Común)
Chip, secuencia de, 162
Chord, 380-384
cHTML (*vea* HTML compacto)
CIDR (*vea* Enrutamiento Interdominios sin Clase)
Cifrado(s), 724
en bloques, 737
modo de, de flujo, 748-749
por sustitución, 727-729
por transposición, 729-730
sólo texto, 727
Circuito Local Inalámbrico (*vea* IEEE 802.16)
Circuito(s)
conmutación de, 147-148
local, 120, 124
virtual(es), 62, 346
concatenados, 422-423
permanentes, 62
subred, 346-347
Clark, David, 46
Clark, Wesley, 51
Clase de Equivalencia de Reenvío, 416
Clases de servicio, IEEE 802.16, 308-309
Clave(s)
algoritmos criptográficos de, simétrica, 737-751
AES, 741-745
DES, 738-741
modos de cifrado, 745-751
Rijndael, 743-745
Chord, 381

Criptografía, 725
depósito de, 820
flujo de, 748
maestras, 784
premaestra, 814
privada, 753
pública, 753
anillo de, 803
certificado, 765-771
infraestructuras de, 768-770
secretas, 753
CLEC (*vea* LEC Competitiva)
Cliente, 4
stub del, 527-529
Clipper, procesador, 820
CMTS (*vea* Sistema de Terminación de Cable Módem)
Codec, 140
Codificación
de forma de onda, 676
entrecomillada imprimible, 598
Manchester diferencial, 274-275
modulación diferencial por, de impulsos, 142
perceptual, 677
Codificación por Desplazamiento de Fase en Cuadratura, 127, 306
Código, 724
firma de, 818
móvil, 817
seguridad, 816-819
polinomial, 196
Código de Autenticación de Mensajes basado en *Hash*, 775, 790
Colisión, 250
Color secuencial con memoria, 694
Comentarios, solicitudes de, 76
Comercio
electrónico, 5
móvil, 11
Comité, borrador de, 75
Comité Nacional de Estándares de Televisión, 694
Compañías Operativas de Bell, 122
Compresión de vídeo, 696-704
algoritmo
de codificación, 696
de decodificación, 696
con pérdidas, 696
JPEG, 697-700
MPEG, 700-704
sin pérdidas, 696
Computadoras, redes de, 2
802.11, 68-71
ARPANET, 50-54
ATM, 61-65
domésticas, 6-9, 23-25
estandarización, 71-77
Ethernet, 65-68
hardware, 14-16
IEEE 802.11, 68-71
inalámbricas, 21-23, 68-71
jerarquía de protocolo, 26-30
modelos de referencia, 37-49
NSFNET, 54-56
orientadas a la conexión, 59-65
software, 26-37
uso, 3-14
X.25, 61
Comunicación
medio de, 5
seguridad en la, 772-785
firewalls, 776-779
inalámbrica, 780-785
IPsec, 772-776
VPNs, 779-780
subred de, 19, 344
Concurso de méritos, 105
Conexión(es)
establecimiento de una, 496-502
TCP, 539-540
liberación de una, 502-506
TCP, 541
persistentes, 652
Confianza, cadena de, 770
Confirmación de Recepción Positiva con Retransmisión, 209
Congestión, control de, 384-396
bit de advertencia, 391
control de fluctuación, 395-396
paquetes reguladores, 391-394
principios, 386-388
subredes de circuitos virtuales, 389-390
subredes de datagramas, 391-395
TCP, 547-548
Conjuntos de Registro de Recursos, 809
Conmutación
de mensajes, 148-149
elementos de, 19
subred de, de paquetes, 20
Conmutadores, 326-328
Ethernet, 281
Conocimiento de los vecinos, 361
Consejo de Actividades de Internet, 75
Consejo de Arquitectura de Internet, 75
Constelación, diagrama de, 128
Consulta recursiva, 588
Contador, modo de, 749-750
Contención, 251
protocolos de, limitada, 261-265
Contenedor o Sobre de Carga Útil Síncrona, 145
Conteo descendente binario, protocolo, 260-261

Control
canal de, común, 161
canal de, de difusión, 161
de flujo basado en tasa, 192
dispositivo de, central, 18, 169
subcapa de, de acceso al medio, 247-342 (*vea también* MAC, subcapa)
Control de Enlace de Datos de Alto Nivel, 234-237
Control Lógico de Enlace, 290-291
Convergencia de transmisión, subcapa, 64
Convertidor Analógico a Digital, 675
Cookie(*s*), Web 626-629
no persistente, 626
persistente, 626
Corporación de Internet para la Asignación de Nombres y Números, 437
Corrección de errores
códigos de, 193
hacia delante, 193-307
Correo
caracol, 588
electrónico, 5, 57, 588-611
agente de usuario, 591-594
alias, 593
armadura ASCII, 598
arquitectura y servicios, 579-591
buzones de, 591
características de entrega, 609-610
codificación base64, 598
codificación entrecomillada imprimible, 598
cuerpo, 591
disposición, 590
encabezados, 595-596
entrega final, 605-611
envío de, 592-593
filtros, 609
formatos de mensaje, 594-602
generación del informe, 590
lectura del, 593-594
MIME, 596-602
perfil de usuario, 593
POP3, 605-608
redacción, 590
SMTP, 602-605
transferencia, 590
transferencia de mensajes, 602-605
visualización, 590
X.400, 589-590
Correo con Privacidad Mejorada, 803-804
Correos, Telégrafos y Teléfonos, administración, 72
CRC (*vea* redundancia cíclica, código de)
Crédito, mensaje de, 522
Criptoanálisis, 726, 750-751
consumo de energía, 751
diferencial, 750-751
lineal, 751
temporización, 751
Criptografía, 724-755
AES, 741-745
clave
pública, 752-755
simétrica, 737-751
criptoanálisis, 726
cuántica, 731-735
DES, 738-741
introducción, 725-727
modos de cifrado, 745-750
principio de Kerckhoff, 726
rellenos de una sola vez, 730-731
Rijndael, 743-745
texto
cifrado, 725
llano, 725
tradicional, 727-730
Criptología, 726
CRL (*vea* Lista de Revocación de Certificados)
Crominancia, 694
CSMA (*vea* Acceso Múltiple con Detección de Portadora)
CSMA/CA (*vea* Acceso Múltiple con Detección de Portadora)
CSMA/CD (*vea* Acceso Múltiple con Detección de Portadora)
CTS (*vea* Libre para Envío)
Cuantización
JPEG, 698
ruido de, 675
Cubeta
algoritmo de
con goteo, 400-403
con *tokens*, 402-405
Cuenta hasta infinito, problema de la, 359-360
Cypherpun k, retransmisores de correo, 821-822

D

D-AMPS, 157-159, 665
Daemen, Joan, 742
Datagramas, 346
servicio de, 33
subredes, 345-347
control de congestión, 391-395
en comparación con circuitos virtuales, 348-350
Datos
entrega de, 302

obsoletos, 658
tramas de, 38
urgentes, 535
David y Goliath, 589
Davies, Donald, 51
DB, 674
DCF, Espaciado Entre Tramas, 299
DCF (*vea* Función de Coordinación Distribuida)
DCT (*vea* Transformación por Coseno Discreto)
DDoS, ataque (*vea* Negación de Servicio Distribuida, ataque)
de facto, estándar, 71
de jure, estándar, 71
Decibel, 89
Demonios, 590
Derechos de autor, 826-828
Derivación(es)
cable de, 272
vampiro, Ethernet, 271
DES (*vea* Estándar de Encriptación de Datos)
Desconocimiento, seguridad por, 726
Descubrimiento de ruta, redes *ad hoc*, 376-379
Desempeño
aspectos del, 557-573
medición del, de las redes, 560-562
Desmultiplexión, 31
Desplazamiento de frecuencia, modulación por, 126
Desprendimiento de carga, 394
Detección
códigos de, de errores, 193
de portadora, supuesto, 250
protocolos de, de portadora, 255
temprana aleatoria, 395
DHCP (*vea* Protocolo de Configuración de *Host* Dinámico)
Diálogo, control de, 40
Diffie-Hellman, intercambio de claves de, 791-792
DIFS (*vea* DCF, Espaciado Entre Tramas)
Difusión, 15, 276, 368
enrutamiento por, 368-370
Ethernet, 279
redes de, 15
tormenta de, 330, 558
Digramas, 728
Dijkstra, algoritmos de la ruta más corta, 353-355
Dirección
bajo su responsabilidad, 463
traducción de, de red, 444-448
Direccionamiento, 31
con clase, 437
Direcciones
IP, 436-438, 441-444
transporte, 493-496
Directivas, HTML, 630
Dirigido por control, MPLS, 417
DIS (*vea* Borrador de Estándar Internacional)
Discos
arreglo de, 708
granja de, 708
Diseño, aspectos de, 30-31
capa
de enlace de datos, 184-492
de red, 343-350
de transporte, 492-513
Disociación, servicio 802.11, 301
Dispersión cromática, 95
Disposición, correo electrónico, 590
Dispositivo de Interfaz de Red, 133
Distorsión, 125
Distribución, servicio 802.11, 301
División, 708
DIX, Ethernet, 275-276, 278
DMCA (*vea* Ley de Propiedad Intelectual para el Milenio Digital)
DMT (*vea* MultiTono Discreto)
DNS (*vea* Sistema de Nombres de Dominio)
seguridad, 809-811
seguro, 809-811
DOCSIS (*vea* Especificación de Interfaz para Servicio de Datos por Cable)
Doctrina de uso, 827
Documento Web
dinámico, 643-651
estático, 623-643
Dominio(s)
de colisión, Ethernet, 282
de nivel superior, 580
Kerberos, 797
reflectometría en el, del tiempo, 272
Dos ejércitos, problema de los, 503-504
DSLAM (*vea* Multiplexor de Acceso de Línea Digital de Suscriptor)
DSSS (*vea* Espectro Disperso de Secuencia Directa)
Dúplex total, 129
Duplexación por División de Frecuencia, 307
Duplexación por División de Tiempo, 307
DVMRP (*vea* Protocolo de Enrutamiento Multidifusión de Vector de Distancia)
DWDM (*vea* Multiplexión Densa por División de Longitud de Onda)

E

E1, portadora, 142
ECB (*vea* Libro de Código Electrónico, modo de)
EDE, modo, DES, 741
EDGE (*vea* Velocidades de Datos para la Evolución del GSM)

EEE, modo, DES, 741
EIFS (*vea* Espaciado Entre Tramas Extendido)
Eisenhower, Dwight, 51
El ataque de cumpleaños, 763-765, 782
Elefantes, apocalipsis de los, 46-47
Emociones, símbolos de, 588
Emoji, 669
Encabezado, 29
  correo electrónico, 591
  de autenticación, 774-775
  Ethernet, 275-276
  paquete
    IPv4, 433-436
    IPv6, 466-469
  predicción de, 568
  segmento TCP, 536-539
  trama, 201-203
Encapsulado de Carga Útil de Seguridad, 775-776
Encolamiento justo ponderado, 409
Encriptación lineal, 751
Enlace
  Bluetooth, 315
  capa de, de datos, 38, 183-246
    control de errores, 192-200
    control de flujo, 192
    cuestiones de diseño, 184-192
    procedimientos de interfaz, 202-204
    protocolo HDLC, 234-237
    protocolo LCP, 239-242
    protocolo NCP, 239, 242
    protocolo PPP, 238-242
    protocolo símplex de parada y espera, 206-211
    protocolo símplex sin restricciones, 204-206
    protocolos, 200-228
    protocolos de ventana corrediza, 211-228
    protocolos elementales, 200-211
    relleno de bits, 190-191
    relleno de caracteres, 189-190
    verificación de protocolo, 229-234
  encriptación de, 723
Enmascaramiento temporal, 677
Enmascarar, 677
Enrutador(es), 19, 326, 328
  adyacente, 457
  de multidifusión, 711-712
  designado, 457
  m, 711
  multiprotocolo, 421
Enrutamiento, 31
  algoritmo de, 20, 347, 350-384
    adaptativo, 351-352
    ARPANET, 357, 454
    Bellman-Ford, 357-360, 454
    estado del enlace, 360-366
    Ford-Fulkerson, 357-360
    host móvil, 372-375
    inundación, 355-357
    IS-IS, 365-366
    jerárquico, 366-368
    multidifusión, 370-372
    no adaptativo, 351
    optimización, 352-353
    OSPF, 454-459
    proporcional, 408
    red *ad hoc*, 375-380
    reenvío por ruta invertida, 369-370
    ruta más corta, 353-356
    vector de distancia, 357-360
  entre redes, 426-427
  estático, 351
  jerárquico, 366-368
  multidestino, 368
  proporcional, 408
Enrutamiento Interdominios sin Clase, 441-444
Entorno de Aplicaciones Inalámbricas, 664
Entrada agregada, 444
Entrelazado, vídeo, 693
Entunelamiento, 425-427
Errores
  control de, 31
  detección y corrección, 192-200
ESP (*vea* Encapsulado de Carga Útil de Seguridad)
Espaciado Corto Entre Tramas, 299
Espaciado Entre Tramas Extendido, 299
Espacio Entre Tramas PCF, 299
Especificación de Interfaz para Servicio de Datos por Cable, 173
Espectro disperso
  802.11, 294-295
  historia, 102
  secuencia directa, 102, 294
Espectro Disperso con Salto de Frecuencia, 102, 294
  tiempo de permanencia, 294
Espectro Disperso de Secuencia Directa, 102, 294
Espectro Disperso de Secuencia Directa de Alta Velocidad, 295
Espectro electromagnético, 100-102
  políticas de, 105-106
Esquema, World Wide Web, 623-625
  ftp, 624
  gopher, 624-625
  http, 623-624
  mailto, 624-625
  news, 624
  rtsp, 684
  telnet, 624-625
  URL, 623
Estación(es), 249
  central, 112, 272, 326-327

modelo de, 249
problema de, expuesta, 269
problema de, oculta, 269
Estado
enrutamiento por, del enlace, 360-366
máquinas de, finito
inicial, 230
modelos, 229
protocolos de espera y parada, 229-232
TCP, 541-543
Estándar
de facto, 71
de jure, 71
internacional, 75
Estándar Borrador, 77
Estándar de Encriptación Avanzada, 741-745
Rijndael, 743-745
Estándar de Encriptación de Datos, 738-741, 751
modo
EDE, 741
EEE, 741
triple DES, 740-741
Estándar Propuesto, 77
Estándares de Firmas Digitales, 758-759
Estandarización, 71-77
Esteganografía, 824-825
Estrella pasiva, 98
Ethernet, 17, 65-68, 271-292 (*vea también* Fast Ethernet, Gigabit Ethernet)
10BASE-F, 273
10BASE-T, 272
cable de derivación, 272
cableado, 271-274
clásica, 327
codificación Manchester, 274-275
conmutada, 281-283, 328-336
delgado, 272
derivaciones vampiro, 271
desempeño, 279-281
difusión, 276
DIX, 67, 275-276, 278
dominio de colisión, 282
Fast, 283-285
formato de trama, 276
Gigabit, 286-290
grueso, 271
historia, 65-67
multidifusión, 276
protocolo, 275-279
repetidor, 274
retroceso exponencial binario, 278-279
retrospectiva, 291-292
Etiquetas
conmutación de, multiprotocolo, 415-417
HTML, 630
Expresión, libertad de, 822-826
Extensión
de portadora, Gigabit Ethernet, 287-288
encabezados de, 469
Extensiones Multipropósito de Correo Internet, 596-602

F

FAQ (*vea* preguntas más frecuentes)
Fase, modulación de, 126
Fast Ethernet, 283-286
4B/5B, 285
8B/6T, 285
100Base-FX, 285
100Base-T4, 285
100Base-TX, 284-285
cableado, 284-286
concentradores, 285-286
conmutadores, 286
dúplex completo, 285
FDD (*vea* Duplexación por División de Frecuencia)
FDDI (*vea* Interfaz de Datos Distribuidos por Fibra)
FDM (*vea* Multiplexión por División de Frecuencia)
FEC (*vea* Clase de Equivalencia de Reenvío)
Felten, Edward, 827
FHSS (*vea* Espectro Disperso con Salto de Frecuencia)
Fibra(s)
hasta la acera, 709-710
hasta la casa, 710
nodos de, 170
óptica, 93-99
dispersión cromática, 95
en comparación con el alambre de cobre, 98-99
en comparación con satélites, 117-118
monomodo, 94
multimodo, 94
solitones, 96
redes de, 97-98
Filtros, correo electrónico, 609
Firewalls, 776-779
filtro de paquete, 777
puerta de enlace, 778
Firmas digitales, 755-765
ataque de cumpleaños, 763-764
clave
pública, 757-759
simétrica, 756-757
compendios de mensaje, 759-762
Fluctuación, 395-396

Flujo, 397
ataque de reutilización de, de claves, 749
audio de, continuo, 679-683
metaarchivos, 680
reproductor de medios, 680-683
servidor *pull*, 681-682
servidor *push*, 682
tipos MIME, 679-680
control de, 31, 192, 506-510
basado en retroalimentación, 192
basado en tasa, 192
especificación de, 407
medios de, continuo, 674
Ford-Fulkerson, algoritmo de enrutamiento, 357-360
Formularios
HTML, 634-638
PHP, 645-646
Web, 634-638
Fotones, 732
Fourier
serie de, 86-87
transformación de, 676
Fragmentación, interred, 427-431
Fragmentos, ráfaga de, 298
Frame relay, 61
Frecuencia, 100
enmascaramiento de, 677
modulación de, 126
FTP (*vea* Protocolo de Transferencia de Archivos)
FTTC (*vea* fibra(s), hasta la acera)
FTTH (*vea* fibra(s), hasta la casa)
Fuerza de Trabajo para la Ingeniería de Internet, 76
Fuerza de Trabajo para la Investigación de Internet, 76
Función de Coordinación Distribuida, 296, 298-299
Fuzzball, 54

G

G.711, PCM, 686
G.723.1, 687
Gatekeeper, H.323, 686
Generación de páginas Web dinámicas en el cliente, 647-651
Generación del informe, correo electrónico, 590
Gigabit Ethernet, 286-290
8B/10B, 289
10 Gbps, 290
1000Base-CX, 288
1000Base-LX, 288
1000Base-SX, 288
1000Base-T, 288
cableado, 288
extensión de portadora, 287-288
modos de funcionamiento, 287
ráfagas de trama, 288
UTP, 288
Gigabits, protocolos, 569-570
Globalstar, 115-116
Gopher, 624
GPRS (*vea* Servicio de Radio de Paquetes Generales)
GPS (*vea* Sistema de Posicionamiento Global)
Granja de servidores, Web, 621-622
Gray, código de, 294
Gray, Elisha, 119
Grupo de Expertos en Películas, 700
Grupo, maestro, 138
Grupo Unido de Expertos en fotografía, 697
GSM (*vea* Sistema Global para Comunicaciones Móviles)

H

H.225, 687
H.245, 687
H.323, 683-689 (*vea también* voz sobre IP)
gatekeeper, 686
Haces reducidos, 112
Hacia arriba, multiplexión, 510
Hamming
código de, 193-195, 307
distancia de, 193
HDLC (*vea* Control de Enlace de Datos de Alto Nivel)
HDTV (*vea* televisión, de alta definición)
Hertz, 100
Hertz, Heinrich, 100
Hipertexto, 612
Hipervínculos, 612
HMAC (*vea* Código de Autenticación de Mensajes basado en *Hash*)
Hojas de estilo, HTML, 634
Hombre de en medio, ataque de, 792
Hosts, 19
móviles, 372
enrutamiento para, 372-375
HTML
compacto
ejemplo, 670
en comparación con HTML 1.0, 668
dinámico, 647
vea Lenguaje de Marcado de Hipertexto
HTTP seguro, 813
Huella, 112
Hz (*vea* Hertz)

I

IAB (*vea* Consejo de Actividades de Internet)
ICANN (*vea* Corporación de Internet para la Asignación de Nombres y Números)
ICMP (*vea* Protocolo de Mensajes de Control en Internet)
IDEA (*vea* Algoritmo Internacional de Encriptación de Datos)
Identificador de nodo, 381
IEEE (*vea* Instituto de Ingenieros Eléctricos y Electrónicos)
IEEE 802.1Q, 333-336
IEEE 802.2, 290-291
IEEE 802.3u, 284
IEEE 802.11, 292-302
- 802.11a, 294-295
- 802.11b, 295
- 802.11g, 295
- capa física, 293-295
- Códigos Walsh/Hadamard, 295
- CSMA/CA, 296-297
- CTS, 297-298
- en comparación con 802.16, 303-304
- Espaciado Corto Entre Tramas, 299
- Espaciado Entre Tramas DCF, 299
- Espaciado Entre Tramas Extendido, 299
- Espacio Entre Tramas PCF, 299
- Espectro Disperso con Salto de Frecuencia, 294
- Espectro Disperso de Secuencia Directa, 294
- Espectro Disperso de Secuencia Directa de Alta Velocidad, 295
- formato de trama, 299-300
- Función de Coordinación Distribuida, 296, 298-299
- Función de Coordinación Puntual, 296, 298-299
- Multiplexión por División de Frecuencias Ortogonales, 294-295
- pila de protocolos, 292-293
- Privacidad Inalámbrica Equivalente, 300
- problema de la estación expuesta, 296
- problema de la estación oculta, 296
- protocolo de subcapa MAC, 295-299
- ráfaga de fragmentos, 298
- RTS, 297-298
- secuencia Barker, 294
- seguridad, 781-783
- servicios, 301-302
  - asociación, 301
  - autenticación, 302
  - desautenticación, 302
  - distribución, 301
  - entrega de datos, 302
  - integración, 301
  - privacidad, 302
  - reasociación, 301
- tiempo de permanencia, 294
- trama de *beacon*, 298
- Vector de Asignación de Red, 297
- WAP, 673

IEEE 802.12, 293
IEEE 802.15 (*vea* Bluetooth)
IEEE 802.16, 135, 302-310
- capa física, 306-307
- clases de servicio, 308-309
- Duplexación por División de Frecuencia, 307
- Duplexación por División de Tiempo, 307
- en comparación con 802.11, 303-304
- estructura de trama, 309-310
- pila de protocolos, 305-306
- seguridad, 307-308
- servicio
  - de mejor esfuerzo, 308
  - de tasa de bits constante, 308
  - de tasa de bits variable en tiempo real, 308
- subcapa MAC, 307-309

IETF (*vea* Fuerza de Trabajo para la Ingeniería de Internet)
If-Modified-Since, encabezado de solicitud, 659
IGMP (*vea* Protocolo de Administración de Grupo de Internet)
IGP (*vea* Puerta de enlace interior, protocolo de)
Igual a igual, 7-9
Igualdad privada, 59
Iguales, 27
IKE (*vea* Intercambio de Claves de Internet)
ILEC (*vea* LEC Obligada)
IMAP (*vea* Protocolo de Acceso a Mensajes de Internet)
I-mode, 665-670
- aceptación en occidente, 667
- cHTML, 668-670
- emoji, 669
- en comparación con WAP, 671
- estructura del software, 667-668
- facturación, 666
- microteléfonos, 667
- modelo de negocios, 666
- pila de protocolos, 672
- servicios, 666
  - oficiales, 666

IMP, 52
IMT (*vea* Unión Internacional de Telecomunicaciones)
IMTS (*vea* Sistema Mejorado de Telefonía Móvil)
Inalámbrica
- fija, 10, 135 (*vea también* IEEE 802.16)
- móvil, 10

Índice, HTML, 633-634
Inetd, 533
Information-mode, 665-670
Infraestructura de Clave Pública, 768-770
Inicio de sesión remota, 57
Instituto de Ingenieros Eléctricos y Electrónicos, 75

Instituto Nacional de Estándares Nacionales, 74
Instituto Nacional de Estándares y Tecnología, 75, 741
Integración, servicios, 802.11, 301
Intercambio de Claves de Internet, 773
Interconectividad, 418-431
  circuitos virtuales, 422-423
  enrutamiento, 426-427
  entunelamiento, 425-426
  fragmentación, 427-431
  local, 322-323
Interconectividad no orientada a la conexión, 423-425
Interfaz, 27
Interfaz de Datos Distribuidos por Fibra, 283
Interfaz de Puerta de Enlace Común, 644-645
Internet, 25, 50-59, 237-242, 431-473, 524-556
  a través de cable, 170-176
    en comparación con ADSL, 175-176
  arquitectura, 58-59
  capa de, 42
  capa de red, 431-473
  demonio de, 533
  direcciones, 436-448
  historia, 50-56
  introducción, 56-58
  multidifusión de, 461-462
  principios de diseño, 431-432
  proveedor de servicios de, 56
  radio en, 683-685
  sociedad de, 77
Interredes, 25-26
Intranets, 59
Intruso, 725
Inundación
  algoritmo, 355, 357
  selectiva, 355
IP, 432-444, 464-473
  móvil, 462-464
  seguridad, 772-776
    encabezado de autenticación, 774-775
    Encapsulado de Carga Útil de Seguridad, 775-776
    modo de transporte, 773
    modo de túnel, 773
IPv4, 432-444
  direcciones, 436-448
    clases A, B, C, D, 437
    con clases, 437
    encabezado, 433-436
    máscara de subred, 439-441
    opciones, 436
    sin clases, 441-444
IPv5, 433
IPv6, 464-473
  controversias, 471-473
  encabezado principal, 466-469
  encabezados de extensión, 469-471
Iridium, 114-116
IRTF (*vea* Fuerza de Trabajo para la Investigación de Internet)
IS (*vea* estándar, internacional)
ISAKMP (*vea* Asociación para Seguridad en Internet y Administración de Claves)
IS-IS (*vea* sistema intermedio-sistema intermedio)
ISM (*vea* bandas industriales, médicas y científicas)
ISO, 74
ISP (*vea* Internet, proveedor de servicios de)
ITU (*vea* Unión Internacional de Telecomunicaciones)
IV (*vea* vector de inicialización)
IXC (*vea* portadora entre centrales)

J

Japanese Telephone and Telegraph Company, 665-670
Java, seguridad de subprogramas de, 817
Java Server Pages, 646
JavaScript, 647-651
  seguridad, 818
Jerarquía Digital Síncrona, 144
JPEG, compresión, 697-700
  codificación de longitud de ejecución, 699
  cuantización, 698-699
  DCT, 698
  preparación del bloque, 697
  vea Grupo Unido de Expertos en Fotografía
JSP (*vea* Java Server Pages)
Juicio Final Modificado, 122
Jumbogramas, 470, 472
JVM (*vea* Máquina Virtual de Java)

K

Karn, algoritmo de, 552
KDC (*vea* Centro de Distribución de Claves)
Kerberos, 796-798
Kerckhoff, principio, 726
Knudsen, Lars, 742

L

Lamarr, Hedy, 102
LAN inalámbrica, protocolo, 265-270
LAN (*vea* red, de área local)
LAN virtual, 328-336
LAP (*vea* Procedimiento de Acceso de Enlace)

LAPB, 234-235
Last-Modified, encabezado, 658-659
LATA (*vea* área(s), de acceso y transporte local)
LCP (*vea* Protocolo de Control de Enlace)
LDAP (*vea* Protocolo Ligero de Acceso al Directorio)
LEC (*vea* portadora, de intercambio local)
LEC Competitiva, 134
LEC Obligada, 134
Leche, política, 394
Lenguaje de Hojas de estilo Extensible, 639-642
Lenguaje de Marcado de Hipertexto, 629-639
- atributos, 630
- celdas, 633
- directivas, 630
- encabezado, 630
- etiquetas, 630
- formularios, 634-638
- hipervínculos, 632-633
- hoja de estilo, 634
- índice, 633-634
- título, 630

Lenguaje de Marcado de Hipertexto Extendido, 642
Lenguaje de Marcado Extensible, 639-642
Lenguaje de Marcado Inalámbrico, 664
Lenguajes de marcado, 629
LEO (*vea* Satélites de Órbita Terrestre Baja)
Ley de Propiedad Intelectual para el Milenio Digital, 827-828
Libre para Envío, 269
Libres de colisiones, protocolos, 259-270
Libro de Código Electrónico, modo de, 745-746
Línea de Fases Alternas, 694
Línea Digital Asimétrica de Suscriptor, 130-134
- en comparación con cable, 175-176

Lista de correo, 591
Lista de Revocación de Certificados, 771
Little endian, máquina, 433
Llamada a Procedimiento Remoto, 526-529
- marshaling, 527-529
- *stub*
  - del cliente, 527-529
  - del servidor, 527-529

LLC (*vea* Control Lógico de Enlace)
LMDS (*vea* Servicio Local de Distribución Multipunto)
Localidad base, 373
Localización, canal de, 161
Localizadores Uniformes de Recursos, 614, 622-625
Longitud de ejecución, codificación de, 699
LTP (*vea* Protocolo de Transporte de Carga Ligera)
Lugar, red de Petri, 232
Luminancia, 694
Luz, velocidad de la, 100

M

MAC, subcapa, 247-342
- asignación
  - del canal, 248-251
  - dinámica de canales, 249-251
  - estática de canales, 248-249
- Bluetooth, 310-317
- conmutación de la capa de enlace de datos, 317-336
- Ethernet, 271-292
- LAN inalámbrica, 292-302
- protocolos de acceso múltiple, 251-270

MACA inalámbrico, 270-271
MACA (*vea* Acceso Múltiple con Prevención de Colisiones)
Macrobloques, 702
MAHO (*vea* Transporte Asistido Móvil de Celda)
MAN inalámbrica (*vea* IEEE 802.16)
MAN (*vea* red, de área metropolitana)
Manchester, codificación, 274-275
MANET (*vea* Redes *ad hoc* Móviles)
Mantenimiento de rutas, redes *ad hoc*, 379-384
Mapa de bits, protocolo de, 259-260
Máquina Virtual de Java, 650, 817
Marca(s)
- aleatorias, 786
- de agua
  - alta, 682
  - baja, 682

Marconi, Guglielmo, 21
MARS, 742
Marshaling, parámetro, 527-529
Máscaras, reproductor de medios, 680
Matsunaga, Mari, 665
Maxwell, James Clerk, 66
Mbone, 711-714
MD5, 760
Medios
- continuos, 674
- de transmisión guiados, 90-99
- físicos, 27

Mensaje(s)
- compendios de, 759-762
  - MD5, 760
  - SHA-1, 761-762
- instantáneos, 7

MEO (*vea* Satélites de Órbita Terrestre Media)
Merkle, Ralph, 755
Metaarchivo, 680
Metcalfe, Bob, 23, 66
Métodos, 652
MFJ (*vea* Juicio Final Modificado)
Michelson-Morley, experimento, 66
Microceldas, 154
Middleware, 2

MIME, tipos, 599-602, 617-618
MIME (*vea* Extensiones Multipropósito de Correo Internet)
Minirranuras, 174
MMDS (*vea* Servicio de Distribución Multipunto y Multicanal)
Mockapetris, Paul, 47
Modelo cliente-servidor, 4-5
Modelo de Referencia OSI de ISO, 37
Módems, 125-130
Modo de Transferencia Asíncrona, 61-65
Modo(s)
 de cifrado, 745-750
 de transporte, IPsec, 773
 de túnel, IPsec, 773
 promiscuo, 319
Modulación, 142
 amplitud, 126
 cuadratura, 127
 delta, 143
 fase, 126
 frecuencia, 126
 QAM, 127
Modulación de Amplitud en Cuadratura, 127, 710
Modulación por Codificación de Impulsos, 140
Modulación por Codificación de Malla, 128
Mosaic, 611
MOSPF, 714
MP3, 676-679
MPEG
 capa de audio 3 de, 676-679
 compresión, 700-704
  MPEG-1, 700-703
  MPEG-2, 700-704
  sincronización de audio/vídeo, 701
  tipos de tramas, 701-702
 vea Grupo de Expertos en Películas
MPLS
 dirigido por control, 417
 orientado a datos, 417
MSC (*vea* Centro de Conmutación Móvil)
MTSO (*vea* Oficina de Conmutación de Telefonía Móvil)
MTU (*vea* Unidad Máxima de Transferencia)
Multiacceso
 canales, 247
 red de, 455
Multidifusión, 15, 276, 370
 Internet, 461-462, 712-714
 enrutamiento por, 370
Multidifusión Independiente del Protocolo, 714
Multidrop, cable, 66
Multimedia, 674-714
 audio, 674-692 (*vea también* audio)
  de flujo continuo, 679-683
  digital, 674-676
 compresión
  de audio, 676-679
  de vídeo, 696-704
 en red, 674-714
 introducción al vídeo, 692-696
 Mbone, 711-714
 MP3, 676-679
 radio en Internet, 683-685
 reproductor de medios, 680-683
 RTSP, 680-683
 servidor de medios, 680-683
 telefonía de Internet, 685-692
 vídeo bajo demanda, 704-711
 voz sobre IP, 685-692 (*vea también* voz sobre IP)
Múltiples trayectorias, desvanecimiento por, 69, 104
Multiplexión, 31, 510-511
 hacia abajo, 511
 hacia arriba, 510
Multiplexión Densa por División de Longitud de Onda, 267
Multiplexión por División de Frecuencia, 137-140
Multiplexión por División de Frecuencias Ortogonales, 294
Multiplexión por División de Longitud de Onda, 138-140
Multiplexión por División de Tiempo, 137, 140-143
Multiplexor de Acceso de Línea Digital de Suscriptor, 134
MultiTono Discreto, 132

N

Nagle, algoritmo de, 545-547
NAP (*vea* Punto de Acceso a la Red)
NAV (*vea* Vector de Asignación de Red)
Navajo, código, 724
Navegador Web, 612, 614-618
 aplicación auxiliar, 617-618
 Mosaic, 611
 plug-in, 616-617
NCP (*vea* Protocolo de Control de Red)
Needham-Schroeder, autenticación de, 794-795
Negación de Servicio Distribuida, ataque, 778
Negociación, 32
NID (*vea* Dispositivo de Interfaz de Red)
NIST (*vea* Instituto Nacional de Estándares y Tecnología)
Nivel(es), 26
 acuerdo de, de servicio, 400
 dominios de, superior, 580
NNTP (*vea* Protocolo de Transferencia de Noticias en Red)
No repudio, 756
Nombres Universales de Recursos, 625
Nombres
 asignación segura, 806-813
 servidor de, 495
  DNS, 586-588

Notación de Sintaxis Abstracta 1, 768
Notación decimal con puntos, 437
Noticias, 57
NSA (*vea* Agencia Nacional de Seguridad)
NSAP (*vea* Punto de Acceso al Servicio de Transporte)
NSFNET, 55
NTSC (*vea* Comité Nacional de Estándares de Televisión)
NTT DoCoMo, 665-670
Nyquist, Henry, 89

O

OFDM (*vea* Multiplexión por División de Frecuencias Ortogonales)
Oficina de Conmutación de Telefonía Móvil, 155
Oficina(s)
  central local, 120
  de arrancado de cinta de papel, 149
  interurbanas, 120
  tándem, 120
Olsen, Ken, 6
Onda(s)
  infrarrojas, 106-107
  longitud de, 100
  milimétricas, 106-107
  transmisión por, de luz, 107-108
ONU (*vea* Unidad de Red Óptica)
Oprimir para hablar, sistema de, 153
Organización de Estándares Internacionales, 74-75
Oryctolagus cuniculus, 28
OSI, modelo de referencia, 37-41
  crítica, 46-48
  en comparación con TCP/IP, 44-46
OSPF (*vea* Protocolo Abierto de Ruta más Corta)
Otway-Rees, protocolo de autenticación, 795-796

P

Páginas Web dinámicas en el servidor, generación de, 643-647
PAL (*vea* Línea de Fases Alternas)
Palabra codificada, 193
Paquete(s), 15
  calendarización de, 408-409
  conmutación de, 150-151, 344
  filtro de, 777
  reguladores, 391-394
PAR (*vea* Confirmación de Recepción Positiva con Retransmisión)
Par trenzado sin blindaje, 91-92
  categoría, 3-5, 91-92
Parada y espera, protocolo, 206-211
Paridad, bit de, 194
PCF (*vea* Función de Coordinación Puntual)
PCM (*vea* Modulación por Codificación de Impulsos)
PCS (*vea* Servicios de Comunicaciones Personales)
PEM (*vea* Correo con Privacidad Mejorada)
Perfil(es)
  Bluetooth, 312-313
  de usuario, correo electrónico, 593
Perl, secuencia de comandos de, 644
Perlman, Radia, 324
Persistencia, temporizador de, 552
Petri, red de, 232
PGP (*vea* Privacidad Bastante Buena)
PHP (*vea* Procesador de Hipertexto)
Piconet, 311
PIFS (*vea* Espacio Entre Tramas PCF)
PIM (*vea* Multidifusión Independiente del Protocolo)
Píxel, 695
PKI (*vea* Infraestructura de Clave Pública)
Plug-in, 616
Polinomio generador, 197-200
POP (*vea* Punto de Presencia)
POP3 (*vea* Protocolo de Oficina de Correos Versión 3)
Portadora
  de intercambio local, 122
  de onda senoidal, 126
  entre centrales, 122
Portadores
  comunes, 72
  hoteles de, 59
Portal, 70
Posición orbital, control de la, 111
POTS (*vea* Servicio Telefónico Convencional)
PPP (*vea* Punto a Punto, Protocolo)
Predicción, codificación por, 143
Preguntas más frecuentes, 610
Presentación, capa de, 41
Primitivas, 34
Principio(s)
  criptográficos, 735-736
    actualización, 736
    Kerckhoff, 726
    redundancia, 735-736
  de diseño, Internet, 431-432
  de optimización, 352-353
Privacidad, 302, 819-820
  amplificación de, 734
Privacidad Bastante Buena, 799-804
  claves, 802
  formato de mensaje, 802
  funcionamiento, 801
  IDEA, 799
Privacidad Inalámbrica Equivalente, 300, 781-783

Procedimiento Avanzado de Control de Comunicación de Datos, 234
Procedimiento de Acceso de Enlace, 234
Procesador de Hipertexto, 645-646
Procesadores de Mensajes de Interfaz, 52
Procesos, servidor de, 495
Producto, cifrado de, 738
Progresiva, vídeo, 693
Propiedad intelectual, 826
Protocolo Abierto de Ruta más Corta, 454-459
Protocolo de Acceso a Mensajes de Internet, 608-609
    en comparación con POP3, 609
Protocolo de Administración de Grupo de Internet, 462, 713
Protocolo de Aplicaciones Inalámbricas (*vea* WAP)
Protocolo de Configuración de *Host* Dinámico, 453-454
Protocolo de Control de Enlace, 239-242
Protocolo de Control de Red, 239, 242
Protocolo de Control de Transmisión, 42, 532-556, (*vea también* TCP)
Protocolo de Control de Transporte en Tiempo Real, 531-532, 687
Protocolo de Control Síncrono de Enlace de Datos, 234
Protocolo de Datagrama de Usuario, 43, 524-532 (*vea también* UDP)
Protocolo de Datagrama Inalámbrico, 664
Protocolo de Enrutamiento Multidifusión de Vector de Distancia, 712-713
Protocolo de Inicio de Sesión, 689-692 (*vea también* voz sobre IP)
Protocolo de Internet (IP), 432-444, 464-473 (*vea también* IPv4 e IPv6)
Protocolo de Mensajes de Control en Internet, 449-450
Protocolo de Oficina de Correos Versión 3, 605-608
    en comparación con IMAP, 609
Protocolo de Puerta de Enlace de Frontera, 549-461
Protocolo de Resolución de Direcciones, 450-452
    ARP gratuito, 463
    proxy, 452
Protocolo de Sesión Inalámbrica, 664
Protocolo de Transacciones Inalámbricas, 664
Protocolo de Transferencia de Archivos, 448, 624
Protocolo de Transferencia de Hipertexto, 41, 623, 651-656
    conexiones, 652
        persistentes, 652
    encabezados
        de mensaje, 654-656
        de respuesta, 654-656
        de solicitud, 654-656
    métodos, 652-654
    uso de ejemplo, 656
Protocolo de Transferencia de Noticias en Red, 624
Protocolo de Transmisión de Control de Flujo, 556
Protocolo de Transmisión en Tiempo Real, 680-683
Protocolo de Transporte de Carga Ligera, 667
Protocolo de Transporte en Tiempo Real, 529-532, 680-683
Protocolo Ligero de Acceso al Directorio, 588
Protocolo Simple de Acceso a Objetos, 642
Protocolo Simple de Internet Mejorado, 465
Protocolo Simple de Transporte de Correo, 602-605
Protocolo(s), 27
    1 (utopía), 204-206
    2 (parada y espera), 206-211
    3 (PAR), 208-211
    4 (ventana corrediza), 211-216
    5 (retroceso n), 216-223
    6 (repetición selectiva), 223
    AODV, 375
    ARP, 450-452
    ARQ, 209-211
    BGP, 459-461
    BOOTP, 453
    CSMA, 255
    CSMA/CA, 296-297
    CSMA/CD, 257-258
    de enlace de datos, ejemplo de, 204-228
    de multidifusión
        DVMRP, 712
        MOSPF, 714
        PIM, 714
        PIM-DM, 714
        PIM-SM, 714
    de transporte, 492
        código C, 518-521
        ejemplo, 513-524
        entidad de transporte, 515-522
        máquina de estados finitos, 522-524
        primitivas, 513-515
    DHCP, 453-454
    DVMRP, 712-713
    Ethernet, 275-279
    FTP, 448
    H.323, 683-689
    HDLC, 234-237
    HTTP, 41, 623, 651-656
    ICMP, 449-450
    IGMP, 462
    IKE, 773
    IMAP, 608-609
    inicial de conexión, 495
    IPv4, 433-444
    IPv6, 464-473
    IS-IS, 365-366
    ISAKMP, 773
    jerarquía de, 26-30
    LAP, 234
    LAPB, 234-235
    LCP, 239-242
    LDAP, 588

LTP, 667
MACA, 269-270
MACAW, 269-270
máquina de, 229
MOSPF, 714
NCP, 239, 242
NNTP, 624
OSPF, 454-459
PAR, 209-211
pila de, 28
802.11, 292-293
Bluetooth, 313-314
H.323, 687
i-mode, 668
IEEE 802.16, 305-306
OSI, 39
TCP/IP, 43
WAP 1.0, 664
WAP 2.0, 672
PIM, 714
POP3, 605-608
PPP, 268-242
RARP, 453
RSVP, 410-412
RTCP, 531-532
RTP, 529-532
RTSP, 680-683
SCTP, 556
SDLC, 234
SIPP, 465
SMTP, 602-605
SOAP, 642
TCP, 532-566
UDP, 524-632
verificación de, 229-234
WAP, 663-665, 670-673
WDMA, 265-267
WDP, 664
Protocolos de Acceso Múltiple por División de Longitud de Onda, 265-267
Protocolos de Control en Internet, 449-454
Proxy, Web, 657-659
Psicoacústica, 677
PSTN (*vea* Red Telefónica Pública Conmutada)
PTT (*vea* Correos, Telégrafos y Teléfonos, administración)
Puentes, 317, 319-322
aprendizaje hacia atrás, 323
con árbol de expansión, 323-325
remotos, 325-326
transparentes, 322-326
Puerta de enlace, 25, 326-328
de aplicación, 778
H.323, 686
interred, 422
protocolo de, exterior, 427, 454
Puerta de enlace interior, protocolo de, 427, 454
Puerto(s), 494, 533
bien conocidos, 533
de origen, 447
Punto a Punto, protocolo, 238-242
Punto a punto, redes, 15
Punto de acceso, 68
Punto de Acceso a la Red, 56
Punto de Acceso al Servicio de Transporte, 494
Punto de Presencia, 58, 122

Q

Q.931, 687
QAM (*vea* Modulación de Amplitud en Cuadratura)
QoS (*vea* calidad, del servicio)
QPSK (*vea* Codificación por Desplazamiento de Fase en Cuadratura)
Qubits, 733

R

Radiotransmisión, 103-104
Ráfagas de trama, Gigabit Ethernet, 288
RAID (*vea* arreglo redundante de discos baratos)
RARP, 453
RAS (*vea* Registro/Admisión/Estado, canal)
RC4, 751, 781, 815
RC5, 751
RC6, 742
Reasociación, servicio, 802.11, 301
Recorrido de árbol adaptable, protocolo, 263-265
Recuperación de errores, multimedia, 681
Recursos
compartición de, 3
reservación de, 405-406
Red Óptica Síncrona, 144-146
Red Telefónica Pública Conmutada, 118-151
Red (*vea* computadoras, redes de)
RED (*vea* detección, temprana aleatoria)
Redacción, correo electrónico, 590
Red(es)
área de, dorsal, 456
de área amplia, 19-21
de área local, 16-17, 317-323
de área metropolitana, 18
inalámbrica (*vea* IEEE 802.16)
de orilla, 460
domésticas, 6-9, 23-25

dorsal de multidifusión, 711-714
hardware de, 14-26
inalámbricas, 21-23
interconexión, 420-422
multiconectadas, 460
privadas virtuales, 779-780
seguridad en, 721-834
Redes *ad hoc* Móviles, 375
Redes de Entrega de Contenido, 660-662
proxy, 662
Redes y Servicios Avanzados, 55
Redundancia cíclica, código de, 196
Reenvío, 350
asegurado, 414-415
expedito, 413-414
por ruta invertida, 369-370
Referencia, modelo de, 37-49
comparación, 444-446
OSI, 37-41
TCP/IP, 41-44
Reflexión, ataque de, 787-790
Regiones, 366
Registro/Admisión/Estado, canal, 687
Registro(s)
autorizado, DNS, 587
de recursos, DNS, 582-586
Rellenos de una sola vez, 730-734
Repetición
ataque de, 794
selectiva, protocolo, 223-228
Repetidor, 274, 326-327
activo, 98
Reproductor de medios, 680-683
Reservación de recursos, protocolo de, 260, 410-412
Resolvedor, 580
Respuesta, encabezados de, 654
Retransmisión, temporizador de, 550
Retransmisores de correo anónimos, 820-821
Retroalimentación
control de flujo basado en, 192
de cifrado, modo de, 747-748
Retroceso
exponencial binario, algoritmo de, 278-279
protocolo que usa, n, 216-223
RFC (*vea* Comentarios, solicitudes de)
768, 525
793, 532
821, 589, 594
822, 421, 589-590, 594-597, 599, 651, 716, 801, 804, 821
903, 453
951, 453
1034, 580
1048, 453
1058, 360
1084, 453
1106, 539
1112, 462
1323, 532, 539, 570
1379, 556
1424, 803
1519, 442
1550, 465
1644, 556
1661, 238, 241
1662, 239
1663, 239-240
1700, 435
1771, 461
1889, 529
1939, 605
1958, 431
2045, 597, 599
2060, 608-609
2109, 626
2131, 453
2132, 453
2141, 625
2205, 409-410
2210, 407
2211, 407
2246, 816
2251, 588
2326, 680, 682
2328, 455
2362, 714
2401, 772
2410, 772
2440, 800
2459, 767
2460, 466
2535, 809, 811
2597, 414-415
2616, 651, 654, 659
2617, 785
2632, 804
2806, 669
2821, 605, 715
2822, 594
2993, 448
3003, 600
3022, 445
3023, 599
3119, 681
3168, 549
3174, 762
3194, 469
3246, 413

3261, 689
3280, 767
Rijmen, Vincent, 742
Rijndael, 743-745, 751
Rivest, Ronald, 302, 751, 754-755, 781
Roberts, Larry, 51
Rondas, 738
RPC (*vea* Llamada a Procedimiento Remoto)
Rrsets (*vea* Conjuntos de Registro de Recursos)
RSA, algoritmo, 753-755
RSVP (*vea* reservación de recursos, protocolo de)
RTCP (*vea* Protocolo de Control de Transporte en Tiempo Real)
RTP (*vea* Protocolo de Transporte en Tiempo Real)
RTS (*vea* Solicitud de Envío)
RTSP (*vea* Protocolo de Transmisión en Tiempo Real)
Ruido, 125
Ruta más corta, 353
enrutamiento, 353-356

S

SA (*vea* seguridad, asociación de)
SAFER+, 784
Salón de conversación, 7
Satélites de comunicaciones, 109-117
en comparación con fibra óptica, 117-118
GEO, 109-113
Globalstar, 115-116
Iridium, 114-115
LEO, 114-116
MEO, 113-114
Teledesic, 116
VSAT, 112-113
Satélites de Órbita Terrestre Baja, 114-116
Satélites de Órbita Terrestre Media, 113-114
Scatternet, 311
Schneier, Bruce, 742
SCTP (*vea* Protocolo de Transmisión de Control de Flujo)
SDH (*vea* Jerarquía Digital Síncrona)
SDLC (*vea* Protocolo de Control Síncrono de Enlace de Datos)
SECAM (*vea* color secuencial con memoria)
Segmentación y reensamble, 65
Segmentos, UDP, 525
Seguir con vida, temporizador de, 552
Segunda generación, teléfonos móviles de, 157-166
Seguridad, 721-834
ActiveX, 817-818
administración de claves públicas, 765-772
algoritmos de clave
pública, 752-755
simétrica, 737-751
asociación de, 773
certificados, 765-771
código móvil, 816-819
comunicación, 772-785
correo electrónico, 799-804
criptografía, 724-736
DNS, 806-811
firewalls, 776-779
firmas digitales, 755-765
inalámbrica, 780-785
802.11, 781-783
Bluetooth, 783-784
WAP, 785
IPsec, 772-776
JavaScript, 818
PGP, 799-803
PKI, 769
problemas sociales, 819-828
protocolos de autenticación, 785-799
SSL, 813-816
subprogramas de Java, 817
VPN, 779-780
Web, 805-819
Semidúplex, 129
Señal a ruido, relación, 89
Señal Síncrona de Transporte, 145
Serpent, 742, 751
Servicio de Distribución Multipunto y Multicanal, 135
Servicio de Mensajes Cortos, 666
Servicio de Radio de Paquetes Generales, 168
Servicio Local de Distribución Multipunto, 135
Servicio Telefónico Convencional, 132
Servicio(s)
basado en clase, 412-415
capa de enlace de datos, 184-192
de datagramas confirmados, 33
de mejor esfuerzo, IEEE 802.16, 308
de tasa de bits
constante, IEEE 802.16, 308
variable no en tiempo real, 308
diferenciados, 412-415
eternidad, 823
integrados, 409-412
no orientado a la conexión, 32-33, 345-347
orientado a la conexión, 32-33, 347-348
primitivas, 34-36
transporte, 481-482
red, 344-345
relación a protocolos, 36-37
Servicios de Comunicaciones Personales, 157

Servidor(es), 4
de directorio, 495
de medios, 680-683
de vídeo, 706-709
arquitectura, 707-708
ejemplo de, de archivos de Internet, 488-492
granja de, 621-622
pull, multimedia, 681
push, multimedia, 682
replicación del, 659-660
Web, 618-622
espejo, 659-660
replicación, 659-660
transferencia TCP, 622
Sesión, 40
clave de, 816
enrutamiento de, 350
Shannon, Claude, 89-90
SIFS (*vea* Espaciado Corto Entre Tramas)
SIG, Bluetooth, 310-311
Símbolo, 127
Símplex, 129
Sincronización, 41
Síncrono Orientado a la Conexión, canal, Bluetooth, 316
SIP (*vea* Protocolo de Inicio de Sesión) (*vea también* voz sobre IP)
SIPP (*vea* Protocolo Simple de Internet Mejorado)
Sistema autónomo, 427, 432, 456-458
Sistema Avanzado de Telefonía Móvil, 154-157
Sistema de Archivos Seguro, 811
Sistema de Nombres de Dominio, 54, 579-588
espacio de nombres, 580-582
falsificación, 806-808
registro autorizado, 587
seguridad, 809-811
servidores de nombres, 586-588
zonas, 586
Sistema de Posicionamiento Global, 114
Sistema de Terminación de Cable Módem, 173
Sistema Global para Comunicaciones Móviles, 159-162
Sistema intermedio-sistema intermedio, 365-366
Sistema Mejorado de Telefonía Móvil, 153
Sistema Universal de Telecomunicaciones Móviles, 167
Sistema(s)
con pérdidas, vídeo, 696
distribuido, 2
sin pérdidas, vídeo, 696
telefónico, 118-151
circuito local, 124-137
estructura, 119-121
multiplexión por división de frecuencia, 137-138
Multiplexión por División de Longitud de Onda, 138-140
Multiplexión por División de Tiempo, 140-143
políticas, 122-123
troncales, 137-143
telefónico móvil, 152-169
primera generación, 153-157
segunda generación, 157-166
tercera generación, 166-169
tráfico entre, autónomos, 459
Sitio Web del libro, 79
Sitios espejo, 660
S/MIME, 804
SMTP (*vea* Protocolo Simple de Transporte de Correo)
SOAP (*vea* Protocolo Simple de Acceso a Objetos)
Sobre, correo electrónico, 591
Socket, 487-492
Sockets seguros, capa de, 813-816
Solicitud Automática de Repetición, 209
Solicitud de Envío, 269
Solicitud, encabezados de, 654
solicitud-respuesta, servicio, 33
Solitones, 96
SONET (*vea* Red Óptica Síncrona)
SPE (*vea* Contenedor o Sobre de Carga Útil Síncrona)
SSL (*vea sockets* seguros, capa de)
STS (*vea* Señal Síncrona de Transporte)
Stub del servidor, 527-529
Subred, 19, 439
circuitos virtuales, 347-348
comparación entre datagramas y circuitos virtuales, 348-350
comunicación, 344
datagramas, 345-347
máscara de, 439
Subprocesos de color, 417
Supergrupo, 138
Superposición, 212
Supervisión, Trama, 235
Sustitución monoalfabética, 728

T

T1, portadora, 140-143, 709
T2-T4, portadoras, 143
Tarifa, 72
Tasa de datos máxima de un canal
Nyquist, 89
Shannon, 89-90
TCM (*vea* Modulación por Codificación de Malla)
TCP (Protocolo de Control de Transmisión), 532-556
administración de temporizadores, 550-553
algoritmo
de Jacobson, 549-550
de Karn, 552
de Nagle, 545-547

control de congestionamiento, 547-548
datos urgentes, 535
encabezado, 535-539
establecimiento de una conexión, 539-540
inalámbrico, 553-555
indirecto, 553-554
liberación de una conexión, 541
máquina de estados finitos, 541-543
modelado de administración de conexiones, 541-543
modelo del servicio, 533-535
política de transmisión, 543-547
radio en Internet, 684-685
segmentos, 535-539
síndrome de ventana tonta, 545-547
transaccional, 555-556
transferencia, 622
TCPA (*vea* Alianza para una Plataforma Informática Confiable)
TCP/IP, modelo de referencia, 41-44
crítica, 48-49
en comparación con OSI, 44-46
TDD (*vea* Duplexación por División de Tiempo)
TDM (*vea* Multiplexión por División de Tiempo)
Teledesic, 116
Telefonía de Internet, 685-692 (*vea también* voz sobre IP)
Teléfono(s)
celular, 154
inalámbrico, 152
móvil, 152
móviles de primera generación, 153-157
Televisión
de alta definición, 694-695
por antena comunal o común, 169-170
por cable, 169-175, 710
Temas sociales, 12-14, 819-828
Temporización, rueda de, 569
Tercera generación, sistema telefónico móvil de, 166-169
Terminador, enlace de datos, 184, 200
Terminal, 249, 686
Terminales de Apertura Muy Pequeña, 112
Texto
cifrado, 725
llano, 725
conocido, 727
seleccionado, 727
Tiempo
continuo, supuesto, 250
de permanencia, 294
protocolos en, real, 574
ranurado, supuesto, 250
TLS (*vea* capa, de transporte, seguridad de)
Token, 67
administración de, 40
red de Petri, 232
TPDU (*vea* Unidad de Datos del Protocolo de Transporte)
TPDUs, procesamiento rápido de las, 566-569
Trabajo, factor de, 727
Tracto vocal, 676
Tráfico
análisis de, 774
modelado de, 399-400
supervisión de, 400
Tramado, 187-191
Trama(s)
B, MPEG, 701, 703
D, MPEG, 701, 703
datos, 184
de confirmación de recepción, 38
encabezado de la, 201-203
I, MPEG, 701-702
información, 235
no numeradas, 235
P, MPEG, 701-702
vídeos, 692
Transceptor, cable de, 272
Transferencia de mensajes, agentes de, 590
Transformación por Coseno Discreto, 698-700
Transición, 232
máquina de estado finito, 229
Tránsito, redes de, 460
Transmisión
inalámbrica, 100-108
líneas de, 19
Transpondedor(es)
satélites, 109
tubo doblado, 109
Transporte
direcciones de, 493-496
entidad de, 482
servicio de, 481-492
primitivas, 483-486
proveedor, 483
punto de acceso, 494
usuario, 483
Transporte Asistido Móvil de Celda, 159
Tres osos, problema de los, 441
Tres vías, acuerdo de, 500-502, 539-540
Trigramas, 728
Triple DES, 740-741, 751
Troncales
de conexión interurbanas, 120
teléfonos, 137-143
Trudy, 732
TSAP (*vea* Transporte, servicio de, punto de acceso)
Twofish, 742, 751

U

UDP (Protocolo de Datagrama de Usuario), 43, 524-532
encabezado, 526

inalámbrico, 553-555
segmentos, 525-526
UMTS (*vea* Sistema Universal de Telecomunicaciones Móviles)
Unidad de Datos del Protocolo de Transporte, 485
Unidad de Red Óptica, 709
Unidad Máxima de Transferencia, 535
Unidades métricas, 77-78
Unidifusión, 15
Unión Internacional de Telecomunicaciones, 72-74, 166
URL autocertificable, 811-813
URL (*vea* Localizadores Uniformes de Recursos)
URL, Web, 614, 622-625
URN (*vea* Nombres Universales de Recursos)
Usuarios móviles, 9-12, 372
UTP (*vea* par trenzado sin blindaje)

V

V.32, bis de módem, 128
V.34, bis de módem, 128
V.90, bis de módem, 130
V.92, bis de módem, 130
Vacaciones, demonio de, 610
VC (*vea* circuito(s), virtual)
Vecinos activos, 379
Vector de Asignación de Red, 297
Vector de Distancia *ad hoc* bajo Demanda, 375-380
Vector de inicialización, 746
Velocidades de Datos para la Evolución del GSM, 168
Ventana
de congestionamiento, TCP, 548-550
emisora, 212
protocolo, corrediza, 211-228
1 bit, 211-214
repetición selectiva, 223-228
retroceso n, 216-223
receptora, 212
síndrome de, tonta, 545-547
Verificación, suma de, 197
Vídeo, 692-711 (*vea también* compresión de vídeo)
bajo demanda, 704-711
red de distribución, 709-711
campo, 693
codificación, 696
compuesto, 695
crominancia, 694
introducción al, digital, 692-696
entrelazado, 693
HDTV, 694-695
luminancia, 694
NTSC, 693-694
PAL, 693-694
parámetros de barrido, 693, 695
progresiva, 693
SECAM, 693-694
trama, 693
Vino, política, 394
Virus, 818-819
Visualización, correo electrónico, 590
VLAN (*vea* LAN virtual)
Vocoder, 158
Voz
humana, 676
sobre IP, 685-692
comparación entre H.323 y SIP, 691-692
establecer la llamada, 687-689
G.711, 686
G.723.1, 687
gatekeeper, H.323, 686
H.245, 687
H.323, 683-689
métodos SIP, 690
números telefónicos de SIP, 689
pila de protocolos, H.323, 687
protocolos SIP, 690
Q.931, 687
RAS, 687
RTCP, 687
RTP, 687
SIP, 689-692
VPN (*vea* red(es), privadas virtuales)
VSAT (*vea* Terminales de Apertura Muy Pequeña)

W

W3C (*vea* World Wide Web Consortium)
WAE (*vea* Entorno de Aplicaciones Inalámbricas)
Walsh, código de, 164
Walsh/Hadamard, códigos, 295
WAN (*vea* red, de área amplia)
WAP, 11, 663-665, 670-673
arquitectura, 665
capa del portador, 664
Capa Inalámbrica de Seguridad de Transporte, 664
emoji, 672
en comparación
con 802.11, 673
con i-mode, 671
Entorno de Aplicaciones Inalámbricas, 664
pila de protocolos, 664, 672
Protocolo de Datagrama Inalámbrico, 664
Protocolo de Sesión Inalámbrica, 664
Protocolo de Transacciones Inalámbricas, 664

seguridad, 785
uso
de XHTML básico, 673
de XML, 664
WAP 1.0, 663-665
arquitectura, 664-665
pila de protocolos, 664
WAP 2.0, 670-673
emoji, 672
en comparación con WAP 1.0, 671-672
en competencia con 802.11, 673
pila de protocolos, 672
XHTML básico, 673
Watermarking, 826
Watson, Thomas, J., 23
WDM (*vea* Multiplexión por División de Longitud de Onda)
WDMA (*vea* Protocolos de Acceso Múltiple por División de Longitud de Onda)
WDP (*vea* Protocolo de Datagrama Inalámbrico)
Web (World Wide Web), 2, 611-673
aglomeración instantánea, 660
cliente, 614-618
cookie, 625-629
correo de, 610-611
desempeño, 662
esquemas, 623-625
formularios, 634-638
hipervínculo, 612
historia, 57-58, 611-612
HTML, 615, 629-639
HTTP, 41, 623, 651-656
i-mode (*vea* i-mode)
inalámbricas, 662-673
segunda generación, 670-673
WAP 1.0, 663-665
WAP 2.0, 670-673
página, 612
panorama de la arquitectura, 612-629
Redes de Entrega de Contenido, 660-662
seguridad, 805-819
amenazas, 805-806
asignación de nombres segura, 806-813
código móvil, 816-819
SSL, 813-816
XML, 639-642
XSL, 639-642
WEP (*vea* Privacidad Inalámbrica Equivalente)
WiFi (*vea* IEEE 802.11)
WLL (Circuito Local Inalámbrico) (*vea* IEEE 802.16)
WML (Lenguaje de Marcado Inalámbrico)
World Wide Wait, 660
World Wide Web (*vea* Web)
World Wide Web Consortium, 612
WSP (*vea* Protocolo de Sesión Inalámbrica)
WTLS (*vea* Capa Inalámbrica de Seguridad de Transporte)
WTP (*vea* Protocolo de Transacciones Inalámbricas)
WWW (*vea* Web)

X

X.25, 61
X.400, 589-590
X.500, 588
X.509, 767-768
XDSL, 130
XHTML, 642-643
XHTML Basic, 673
XHTML (*vea* Lenguaje de Marcado de Hipertexto Extendido)
XML (*vea* Lenguaje de Marcado Extensible)
XSL (*vea* Lenguaje de Hojas de estilo Extensible)

Z

Zimmermann, Phil, 799
Zipf, ley de, 706
Zona, 686
DNS, 586
prohibida, 499-500

ACERCA DEL AUTOR

Andrew S. Tanenbaum tiene una licenciatura en ciencias del M.I.T. y un doctorado en filosofía por la University of California en Berkeley. Actualmente es profesor de ciencias de la computación en la Vrije Universiteit de Ámsterdam, Países Bajos, donde encabeza el Grupo de Sistemas de Computadoras. También es decano de la Escuela Avanzada de Computación y Procesamiento de Imágenes, una escuela interuniversitaria de posgrado en la que se llevan a cabo investigaciones sobre sistemas paralelos avanzados, distribuidos y de procesamiento de imágenes. No obstante, hace un gran esfuerzo para no convertirse en burócrata.

También ha efectuado investigaciones sobre compiladores, sistemas operativos, conectividad de redes y sistemas distribuidos de área local. Sus investigaciones actuales se enfocan principalmente en el diseño e implementación de sistemas distribuidos de área amplia en la escala de millones de usuarios. Estas investigaciones, realizadas junto con el profesor Maarten van Steen, se describen en *www.cs.vu.nl/globe*. Estos proyectos de investigación han conducido a más de 100 artículos evaluados por expertos en publicaciones periódicas y actas de congresos, así como a cinco libros.

Por otra parte, el profesor Tanenbaum ha producido una cantidad considerable de software; fue el arquitecto principal del *Amsterdam Compiler Kit*, un conjunto de herramientas de amplio uso para escribir compiladores portátiles, y de MINIX, un pequeño sistema operativo tipo UNIX para cursos de sistemas operativos. Este sistema fue la inspiración y base en la que se desarrolló Linux. Junto con sus estudiantes de doctorado y programadores, ayudó a diseñar Amoeba, un sistema operativo distribuido de alto desempeño basado en micronúcleo. MINIX y Amoeba ahora están disponibles gratuitamente en Internet.

Los estudiantes de doctorado del profesor Tanenbaum han cosechado grandes triunfos tras recibir sus títulos. Él está muy orgulloso de ellos.

El profesor Tanenbaum es socio del ACM, miembro senior del IEEE, miembro de la Academia Real Holandesa de Ciencias y Artes, ganador en 1994 del premio Karl V. Karlstrom del ACM para educadores sobresalientes, ganador en 1997 del Premio ACM/SIGCSE por contribuciones sobresalientes en la educación sobre ciencias de la computación y ganador en 2002 del premio Texty por su excelencia en libros de texto. También está listado en Web en el *Who's Who in the World*. Su página principal de World Wide Web se encuentra en *http://www.cs.vu.nl/~ast/*.